Radboud University Nijmegen

Faculty of Social Sciences

Snorkeling for Beginners: Applying Data Programming to Product Moderation in E-commerce

A case study on enabling domain experts to quickly create and improve data programming-based classifiers for the detection of inappropriate products online

Thesis MSc Artificial Intelligence

Author:
Justine Winkler, BSc

Supervisor:
Prof. Dr. Martha Larson

Second reader:
Dr. Iris Hendrickx

External supervisor:
Bas van Berkel, MSc

November 18, 2020

Abstract

Machine learning classifiers can help product moderators on an e-commerce platform by suggesting potentially inappropriate products. To enable rapid reactions to sudden unforeseen events, it is necessary that such classifiers can be quickly created and adjusted. For this purpose, we investigate the usefulness of *data programming*. Data programming enables domain experts (in our case product moderators) to rapidly create an improve classifiers. Domain experts design *labeling functions* (LFs) that implement weak supervision signals (e.g. heuristics) which can generate noisy suggestions of a label (e.g. inappropriate) for single data points. These LFs are combined to train a probabilistic label model, yielding probabilistic labels for a previously unlabeled training data set. With these probabilistic labels, a classifier is trained. This classifier can be steered by adjusting the LFs and thereby the automatic creation of training data. In a case study, we apply data programming to train classifiers for six different categories of inappropriate products. Our LFs need to be designed and improved by non-coder domain experts in a short amount of time. We investigate characteristics of LFs that are based on the domain experts' domain knowledge. Our results show that defining a high-coverage set of LFs or individual high-accuracy LFs for positive cases is challenging for domain experts. Furthermore, not all LFs have a positive effect on the label model's performance. The performance of the trained classifier reflects the domain expert's level of experience with a category.

To improve the classifier in a short amount of time, we investigate the use of intelligently drawn sets of data points, called *inspiration sets*. The purpose of inspiration sets is to inspire the domain experts in improving the LFs. We compare three types of inspiration sets: Previously unlabeled data, data on which LFs previously disagreed and data most mistaken by the classifier. Overall, we show that inspiration sets can trigger substantial performance improvements in a short amount of time. We provide a detailed observation of how these adjustments affect the LFs' effect on the label model's performance. Altogether, our results indicate that data on which LFs conflicted is most promising as an inspiration set, but that the usefulness of inspiration sets varies across monitors.

Acknowledgements

I give a great thank to my first supervisor Prof. Dr. Martha Larson for her continuous support throughout this work and her excellent feedback.
Furthermore, I would like to thank my external supervisor Bas van Berkel for constantly motivating and challenging me and for his precious guidance and feedback.
Overall, conducting this thesis was a great experience. I would like to thank my team at bol.com for an awesome work atmosphere, critical reflections on my ideas, for great team events and for practicing Dutch with me. In the same vein, I would like to thank the assortment quality team at bol.com for their great time investment that made my experiments possible.
Furthermore, I would like to thank Simon Brugman for helpful discussions on data programming and for his valuable feedback on my ideas. Also, I would like to thank Dr. Iris Hendrickx for being my second reader.
Last but not least, I would like to express my great appreciation for my family and friends for always being there for me throughout this work.

Thank you all so much!
Justine Winkler

Contents

Acknowledgements **iii**

1 Introduction **1**
1.1 Motivation for product moderation . . . 2
1.2 Automatically detecting illicit online sales . . . 3
1.3 Motivation for using data programming . . . 4
1.4 Research Questions . . . 6
1.5 Contributions . . . 7
1.6 Outline . . . 8

2 Related Work **9**
2.1 Data programming . . . 9
2.1.1 Data programming application characteristics . . . 10
2.1.2 Conventional LF definition procedure . . . 12
2.1.3 Characteristics of user-defined LFs . . . 12
2.1.4 Improving the LF creation procedure . . . 13
2.1.4.1 Automatic LF generation . . . 13
2.1.4.2 Human-in-the-loop approaches to LF debugging . . . 14

3 Requirements Engineering **17**
3.1 Main principles . . . 17
3.2 Main requirements . . . 18
3.2.1 Functional requirements . . . 18
3.2.2 Nonfunctional requirements . . . 19
3.3 Workflow and main limitations . . . 19
3.4 Conclusion . . . 21

4 Creating Monitors **23**
4.1 Design of the monitor pipeline . . . 24
4.2 Methods . . . 25
4.2.1 Categories . . . 25
4.2.2 Datasets . . . 27
4.2.3 Monitor creation procedure . . . 29
4.2.4 Features . . . 29
4.2.4.1 Textual features . . . 29
4.2.4.2 Categorical attribute features . . . 30
4.2.4.3 Numerical attribute features . . . 31

4.2.5 Classifier . . . 32
4.2.6 Evaluation procedure . . . 32
4.3 Results . . . 33
4.3.1 Number of labels per sample . . . 33
4.3.2 Probability of the label *inappropriate* . . . 35
4.3.3 LFs' statistics and effects on the label model . . . 36
4.3.4 Feature group ablation analysis . . . 38
4.4 Discussion . . . 40

5 Improving Monitors **43**
5.1 Methods . . . 45
5.1.1 Drawing inspiration sets . . . 45
5.1.2 Monitor improvement procedure . . . 46
5.1.3 Evaluation procedure . . . 48
5.2 Results . . . 49
5.2.1 Adjustments inspired by previously unlabeled data . . . 49
5.2.1.1 Number of labels per sample . . . 49
5.2.1.2 Probability of the label inappropriate . . . 51
5.2.1.3 LFs' statistics and effects on the label model . . . 52
5.2.2 Adjustments inspired by data on which LFs previously disagreed 55
5.2.2.1 Number of labels per sample . . . 55
5.2.2.2 Probability of the label inappropriate . . . 56
5.2.2.3 LFs' statistics and effects on the label model . . . 57
5.2.3 Adjustments inspired by data most mistaken by the classifier . . 60
5.2.3.1 Number of labels per sample . . . 60
5.2.3.2 Probability of the label inappropriate . . . 61
5.2.3.3 LFs' statistics and effects on the label model . . . 62
5.2.4 Performance on the test data . . . 65
5.3 Discussion . . . 66

6 General Discussion **70**

7 Future Work **73**

8 Conclusion **75**

1 | Introduction

Machine learning classifiers can help product moderators on e-commerce platforms by suggesting items that might be inappropriate. Since product moderation is a fast-paced domain, it should be possible to quickly create and adjust such classifiers. In this thesis, we present our findings and insights from a case study on a large Dutch e-commerce platform. We explore the application of a method (*Snorkel* [23], based on the *data programming* [26] paradigm) that enables product moderators to quickly steer classifiers independently of technical experts by tweaking the automatic creation of labeled training data.

With the rise of e-commerce platforms over the past decade, online shopping has become a fixed part of our daily life. We are now able to buy whatever we want within minutes from the comfort of our homes.
Bol.com[1] is an e-commerce platform for the Netherlands and Belgium. According to a press statement released in 2019 [5], bol.com has more than 9 million customers. Being a platform denotes that different sellers can upload offers: The online shop features offers by bol.com itself, but also those of partners that can sell products there. There are more than 20.000 of such partners and more than 17 million products on the platform [5]. Both numbers are constantly increasing, with a broad range of new product offers being uploaded every day.
In September 2019, bol.com published a new platform policy on assortment quality [6]. This policy aims to guarantee a certain quality standard across products offered on the platform. To be an appropriate offer on the platform, a product must fulfill the following four criteria:

1. It complies with the law of the country in which it will be offered (Belgium or the Netherlands).
2. Neither its production nor its use is harmful (e.g. to endangered species or to the customer).
3. The product fulfills the expectations of the customer (e.g. in quality).
4. The product is suited for being sold on a platform (e.g. not a service offering).

In the present work, we focus on products that are inappropriate because they violate one of the first two criteria. We focus on these criteria because they were the main area of expertise of the domain experts with whom we closely collaborated in our study. Each of these criteria gives rise to concrete categories of inappropriate products. For

[1] https://www.bol.com

instance, items made of real fur, whose production harms animals, form one category of inappropriate products (arising from the second criterion). Such categories of inappropriate products comprise products from various subsets of the platform's online catalogue and are characterized by different product properties.
To ensure that the products on the platform comply with the platform policy, bol.com has an assortment quality team that monitors all product offers. This team comprises product moderators who specialize in certain categories of inappropriate products. Being specialized in such a category entails two responsibilities. First, the product moderator knows about characteristics of product offers on the platform that allow them to detect products that are inappropriate because they belong to this category (e.g. via characteristic keywords). Second, once a potentially inappropriate item is found, the product moderator is responsible for deciding whether that item is really inappropriate (i.e. belongs to the category) or not. To detect inappropriate products as quickly as possible, the product moderators constantly check the platform for products belonging to their specialism categories. Manually performing this check for each product and each category is not feasible: The large number of products on the platform and the variety of categories of inappropriate products paired with the limited human resources available make it challenging to keep up with the product moderation task. Consequently, the team needs a customized solution to support them in this task so the product moderators can focus on the items that really their require their attention and not miss any inappropriate products.

1.1 Motivation for product moderation

In this section, we motivate the general importance of the product moderation problem at bol.com. We also motivate why the product moderators need an automatized human-in-the-loop solution.

Inappropriate products may hurt the customer, other species or the environment in various ways. Depending on the category they belong to, inappropriate products have different negative outcomes. Some categories contain products that are restricted by law. The products inside these categories can be illegal (e.g. drugs, weapons), not complying with laws of the European Union, only allowed to be sold under conditions that are not met (e.g. medicine being sold without a license) or otherwise restricted by law (e.g. erotic assortment). Other categories contain harmful products. These inappropriate products can be dangerous goods (e.g. low-quality electronics), items that may constitute a health risk (e.g. soft drug paraphernalia), products that only have a criminal purpose, products from child labor or products whose production involves animal cruelty. The longer an inappropriate product stays on the platform, the more damage it can potentially cause. Hence, inappropriate products should be removed as quickly as possible.
With regard to the large number of products on the platform, the requirement for a fast reaction in product moderation makes manual checking unfeasible. The human resources are limited, both in the number of product moderators available and in the amount of information the human attention can process at a time. Several time-consuming aspects of the product moderation task can in fact be carried out without requiring the product moderators' attention. For example, one can directly filter out the typically large number of products that are identified as appropriate and have not changed since the last check. To enable the product moderators to direct their focus

towards items that really require their attention, those aspects of their task that do not need to be carried out manually should be automatized.
Once a potentially inappropriate item has been detected, the next step is to decide whether the product really violates the platform policy. If so, the assortment quality team needs to react. The severity of the violation determines which reaction is adequate. Sometimes, pointing out the mistake to the seller might suffice. In other cases, the team needs to apply strict measures (e.g. taking a product offline). With consideration of the potential implications of such decisions, it is evident that the instance that forms these judgements bears a great deal of responsibility.
Consequently, the assortment quality team needs to be in control of the final judgement on every potentially inappropriate item. Due to this requirement, a fully automatized solution of the product moderation process is not suitable to them. Instead, they need a solution that keeps the human in the loop.

1.2 Automatically detecting illicit online sales

Previous studies have addressed the task of automatically detecting illicit sales online. While we can build on these studies, we cannot use their approaches to tackle the product moderation problem at bol.com. In this section, we review previous approaches and explain why they are not suitable for our problem.

One group of studies aimed at identifying the promotion of illegal sales of wildlife products [39] or controlled substances [18, 19] on Twitter[2]. They followed the same general approach: First, they used a keyword search to collect a dataset of tweets. Then, they employed a Biterm Topic Model to group the tweets according to topics. They proceeded with the groups of tweets whose topic they considered related to online sales. By manually examining these tweets, they successfully identified some trading websites or individual sellers that illegally sold the respective items online. A clear limitation of the above-mentioned approaches is that they rely on unsupervised techniques which do not allow experts to easily input their domain knowledge (which is part of the nonfunctional requirements of a suitable solution for our problem, as described in Chapter 3). It is important to tune parameters like the number of topic groups, making it difficult for domain experts to use these approaches without help from technical experts (which is also part of the nonfunctional requirements). Further, manually investigating the remaining candidates, which are mostly false positives (e.g. [39]) is time-consuming. Also, social media posts (especially tweets) are limited in size, whereas text in the data used for the present approach greatly varies in length. Another limitation of the approaches is that they fully focus on text, whereas attributes of products can be highly indicative in product moderation on an e-commerce platform.
The Invasive Species Internet Monitoring System (Suiter and Sferrazza, 2007 [33]) supports domain experts in identifying websites that trade invasive vertebrate species. Such websites are automatically searched with keyword queries. These queries are based on keywords such as the name of a species or terms that are indicative of sale promotions. A web interface displays the websites that were found and lets the domain experts evaluate each website (e.g. by marking it as relevant or irrelevant). The system that the product moderators at bol.com were working with when we started our study was based on similar queries. A limitation of such a system is that it cannot

[2]https://twitter.com

generalize beyond what is described in the queries. Designing queries that did not miss any inappropriate products while returning as little false positives (i.e. appropriate items flagged as inappropriate) as possible was too time-consuming for the product moderators at bol.com (see Chapter 3).
Arnold, Wartner and Rahm (2016 [2]) proposed a four-step workflow to identify product imitations (e.g. fake brand shoes) on e-commerce platforms: In the first step, product offers that fall into the scope of the problem investigated are queried to form a dataset. The second step is a preprocessing step, in which relevant information is extracted from the offers and standardized into a certain format. In the third step, the data samples are clustered according to the product that is offered in order to simplify comparing offers of the same product. In the last step, they determine how suspicious each offer is using a scoring function. This scoring function is defined over characteristic product attributes (e.g. whether the price is significantly lower than usual for the product). The workflow produces a set of product offers that are likely to be suspicious and need to be manually checked by domain experts. A limitation of this approach is that it requires a good definition of a scoring function. The product moderation task we tackle comprises a high variety of categories of inappropriate items. Defining a good scoring function for each category may be difficult and also not intuitive for product moderators (who are domain experts, but not technical experts). Further, inappropriate products belonging to the same category (e.g. illegal wildlife products, which can occur in various items within the product catalog) can have a broad range of characteristics, so a clustering approach does not appear feasible. In consequence, the last two steps of the workflow are not easily applicable to the problem at hand.

Altogether, previous work on automatically detecting illicit online sales is not directly employable for the problem at hand, since it often relied on techniques which do not comply with our framework conditions (see Chapter 3).

1.3 Motivation for using data programming

In this work, we use the *data programming* paradigm to train classifiers for product moderation tasks without hand-labeled labeled. Data programming automatically generates labeled training data from user-provided *weak supervision* sources. In this section, we briefly introduce weak supervision and the data programming paradigm (a detailed review of related literature can be found in Chapter 2). Then, we motivate our decision to apply this paradigm to the product moderation task.

Weak supervision sources are imperfect or noisy supervision sources (e.g. heuristics, specific knowledge bases) that can be used to label large datasets [26]. *Data programming* leverages multiple user-provided weak supervision sources (which may overlap and conflict) to create training labels [26]. In data programming, weak supervision sources are formulated as functions that suggest a label for a data point (or abstain). These functions are referred to as *labeling functions* (LFs). Data programming applies LFs to unlabeled data, making it possible to observe their agreements and disagreements. The LFs' suggested labels for these data points are then used to learn a probabilistic *label model*. This label model estimates the LFs' accuracies and correlations. Intuitively, a LF's accuracy reflects how trustworthy that LF is. After learning the label model, the LFs' suggestions on the data points can be re-weighted according to their estimated accuracies and correlations. Combining the LFs' re-weighted suggestions produces

probabilistic labels that indicate the probability for each possible label for each data point. These probabilistic labels can be used to train a classifier.

In this thesis, we report findings and insights from a case study carried out at bol.com. At the beginning of our work, we interviewed experts from bol.com's assortment quality team and identified requirements and challenges for a system that could support them in their product moderation task (Chapter 3). Importantly, such a system needed to keep the product moderator in the loop (as we argue in Section 1.1). Also, product moderators needed to be able to react quickly to unforeseen events. One major requirement was that product moderators could set up the system (through expressing their domain knowledge) independently of technical experts so that it would be ready to scan for a certain category of inappropriate products. In system that the product moderators were working with when we started our project, they formulated their domain knowledge by writing a Boolean query. Based on the challenges they experienced with that system (see Section 3.3), the domain experts determined that a new system should be able to generalize beyond their formulated domain knowledge. Also, the system's output should enable domain experts to rank items according to their suggested probability of being inappropriate. All these aspects suggested the usage of a supervised machine learning (ML) classifier.

Supervised ML classifiers need to be trained using labeled training data. In [29], Roh, Heo & Euijong provide an overview of existing approaches to collect training data. They illustrate a scenario that is similar to the present problem: A data scientist aims to create a new ML-based application to control a factory's product quality. That application's task is the classification of products into categories ("defect" or "ok"). Once new types of products or defects occur, a labeled training set is wanted, but hand-labeling might be too expensive. Using this scenario as a running example, the authors provide guidelines for informed decisions on when to employ which data collection method. Figure 1.1 contains a decision flow chart that illustrates these guidelines. We highlighted the conditions and decisions that apply to the present problem in the decision flow chart. In our case, the dataset for a given category was a subset of all products on the platform.

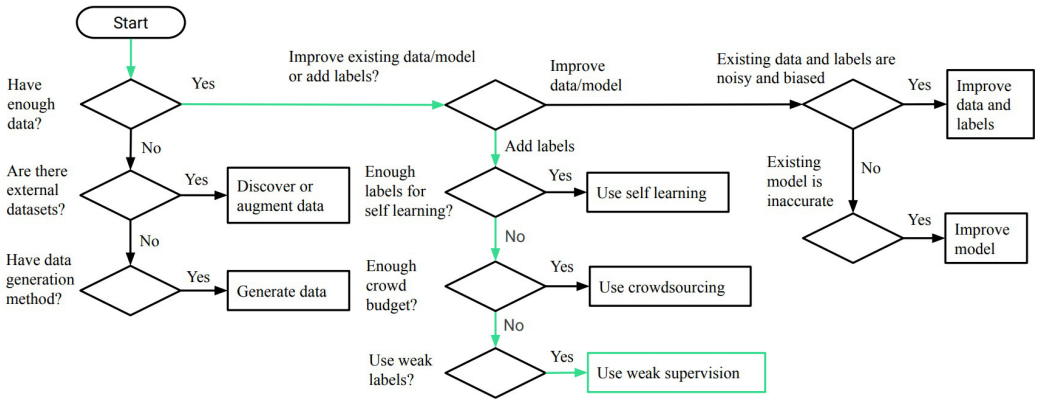


Figure 1.1: A decision flow chart suggesting when to choose which data collection technique (taken from [29]). Conditions and decisions that apply to our case are highlighted in green.

Since our problem was closely tied to bol.com, using external datasets was not suitable. Also, there was no suitable data generation method available. At the start of our research project, we did not have an existing model for a given category. As the dataset for such

a category was unlabeled, the goal was to add labels, but self learning was not feasible. Due to privacy restrictions and our requirement that product moderators should be the ones who define the systems' underlying logic for that category, using crowdsourcing was not an option. Consequently, we decided to employ weak supervision. We further decided to rely on the data programming framework, since it enabled an easy integration of weak supervision sources.

1.4 Research Questions

In this work, we present findings and insights we obtained from a case study on using data programming for our product moderation task. With these findings and insights, we aim to inform the design of a system that is well-suited for our task. We design a prototype of a data programming-based pipeline for such a system (Section 4.1). We call one instance of this pipeline that has been specialized to detect items belonging to a certain category of inappropriate products a *monitor* for the category. In the pipeline, product moderators are the users who aim to steer a classifier for a category by tweaking the generation of training data. They *design* LFs by writing rules (which are then translated into LFs). Leveraging these LFs, a label model generates labeled training data. This data is then used to train the classifier. Using our pipeline, we ask the research questions below.

The purpose of our first research question is to generally understand our LFs. Ours might be different, because our problem setting differs from that in previous data programming-based applications in three ways (see Chapter 4): First, we have users without coding experience (*non-coder domain experts*) that design the LFs by writing rules. Second, our users need to rely on their domain knowledge in defining an initial set of LFs for a category. Third, the users need to create this set of LFs in a short amount of time.
To inform the design of components that may assist the users in their initial definition of LFs for a category (e.g. LF templates), we need insights about the characteristics of LFs that the users define in our problem setting. Furthermore, it might be beneficial to know how such LFs affect the performance of the label model in the data programming pipeline. To the best of our knowledge, this effect has not been studied for such LFs in practice. We ask the following research question:

> **RQ1:** ***What are the characteristics of single LFs that were designed by domain experts in a short amount of time for a high-recall problem and based on their domain knowledge, and what is their effect on the label model's performance?***

The purpose of our second research question is to see how well our whole pipeline performs on a category when trained using these LFs. This performance is measured via the performance of the classifier that is trained in the last step of our pipeline. This way, we check whether a main nonfunctional requirement for a system for our problem (see Chapter 3) is met. This requirement states that the pipeline can be quickly set up for a category and reach a reasonable level of performance. We ask the following research question:

> **RQ2:** ***How does our monitor pipeline perform when a monitor is quickly created by a domain expert based on their domain knowledge for a high-recall problem?***

The purpose of our third research question is to evaluate how useful intelligently drawn sets of data points are in inspiring LF adjustments in a short amount of time, and how these adjustments affect the label model. A main nonfunctional requirement for a system in our problem setting is that the system can be quickly adjusted to improve the performance on a certain category. We focus on attaining such improvements by solely adjusting the initial sets of LFs for a category (see Chapter 5). Our nonfunctional requirements implicate that our users need to make these LF adjustments themselves. We build on previous work that used different strategies to draw data points to inspire LF improvements. Here, we refer to sets of such data points as *inspiration sets*.
We use three different strategies to draw three types of inspiration sets. Two of them (*previously unlabeled data* and *data on which LFs previously disagreed*) are unsupervised, since they do not require gold labels to be drawn. The third (*data most mistaken by the classifier*) is supervised. We ask the product moderators to adjust a category's monitor's initial set of LFs according to each inspiration set in a short amount of time. Note that by "adjusting the LFs", we mean that the product moderators improve the rules that were translated into the initial sets of LFs. These improved rules are again translated into *adjusted* LFs. Valid adjustments to an LF set are creating new LFs or changing or deleting existing ones. We ask the following three research questions:

> **RQ3a:** ***How do LF adjustments inspired by fast inspection of previously unlabeled data change the LFs' effect on the label models' performance?***
>
> **RQ3b:** ***How do LF adjustments inspired by fast inspection of data on which LFs previously disagreed change the LFs' effect on the label models' performance?***
>
> **RQ3c:** ***How do LF adjustments inspired by fast inspection of data most mistaken by the classifier change the LFs' effect on the label models' performance?***

The purpose of our fourth research question is to recommend one specific way to quickly improve the whole pipeline on a given category. We aim to find out which inspiration set type should be used by a system at bol.com. We ask the following research question:

> **RQ4:** ***Which of our three inspiration set types is best-suited for improving the performance of our whole pipeline in a short amount of time?***

1.5 Contributions

We carry out an exploratory case study. With the results of this case study, we seek to inform the design of a data programming-based system for product moderation. The following are our main contributions:

- We analyze the LFs designed by the domain experts at bol.com and provide detailed descriptions of their characteristics and effects. These LFs are different from those in previous work because they were designed by domain experts
  - based on their domain knowledge only
  - for a high-recall problem
  - in a short amount of time
- We investigate the usefulness of fast inspection of three different types of inspiration sets in inspiring LF adjustments. We compare the inspiration sets based on the performance of our whole data programming-based pipeline. We also analyze

the LF adjustments and provide detailed descriptions of how they affect the label model. Guiding users during LF debugging is still an open research area, in which few exploratory studies have been conducted (e.g. [9]).

Altogether, this study teaches a lesson on *Snorkeling* for beginners: It provides insights that may be valuable for others who apply data programming (or the data programming end-to-end system *Snorkel* [23]) to a certain type of problem for the first time.

1.6 Outline

The reminder of this thesis is structured in the following way:

- **Chapter 2** In this chapter, we present the theoretical framework we adopt in this work, i.e. the data programming paradigm. We elaborate on characteristics of applications that used this paradigm and on key
- **Chapter 3** In this chapter, we present the main concepts, requirements and challenges that we identified during initial requirements engineering interviews. These findings were the main drivers in our design of a prototype for a system pipeline.
- **Chapter 4** In this chapter, we present a prototype of a data programming-based pipeline for our product moderation task. We conduct an experiment on quickly setting up this prototype for six different categories of inappropriate items and investigate our research questions RQ1 and RQ2.
- **Chapter 5** In this chapter, we conduct an experiment on quickly improving our prototype for the categories we used in Chapter 4 using our three inspiration set types. We investigate our research questions RQ3a, RQ3b, RQ3c and RQ4.
- **Chapter 6** In this chapter, we provide a general discussion of our case study, explicitly reflecting on limitations.
- **Chapter 7** In this chapter, we point out directions for follow-up work.
- **Chapter 8** In this chapter, we summarize our main findings and insights.

2 | Related Work

2.1 Data programming

Supervised ML-based applications typically require a large amount of labeled training data to obtain the desired level of performance. Creating a hand-labeled training set is expensive and time-consuming [26]. Detailed labeling instructions and teaching people (often domain experts) to properly follow them is required. As such, it may take months until all data points are labeled [4]. Furthermore, in practice, domain experts' time is limited, and labeling data might be regarded as a task of lower priority [32].
A more convenient way to obtain training labels is to rely on weak supervision sources, which are usually cheap to obtain. Here, weak denotes that these sources can be noisy (i.e. make errors) [23]. One type of weak supervision is distant supervision, which uses heuristics to apply knowledge from an existing database to training samples (e.g. [20]). Another type is crowdsourcing, where the labeling task is outsourced to a great number of non-experts (the crowd), who are typically users of an online platform [1]. Other example types of weak supervision are expert-defined pattern matching scripts or domain heuristics [4].
In 2016, Ratner et al. [26] put forward data programming, a ML paradigm in which multiple weak supervision sources are leveraged to programmatically obtain training labels for unlabeled data samples. Importantly, the outputs of these sources are allowed to overlap or even conflict on some data samples. In data programming, the aim is to learn these sources' accuracies and dependencies by training a probabilistic label model on their outputs on the unlabeled dataset. After subsequently combining their outputs, a ML classifier can be trained.
In 2017, Ratner et al. presented Snorkel [23], a framework that applies the data programming paradigm. Accordingly, Snorkel does not involve any hand-labeling by human annotators. Snorkel's pipeline operates in three steps:

1. First, for each weak supervision source, users define a labeling function (LF). An LF is a function (typically written in Python) that takes a single data point as input and either returns a label for that data point or abstains. As an illustration, Listing 2.1 contains a toy heuristic LF expressing that the brand 'steiff' was an indicator for appropriate products.

2. In the second step, these LFs are applied to the data, resulting in a label matrix that contains all LFs' votes (i.e. outputs) on all samples. Due to their potentially noisy nature, taking an unweighted combination of the LFs' votes to obtain labels for the samples is not ideal. To address this issue, Snorkel trains a probabilistic label model on the label matrix. This model learns estimations of the LFs' correlations

[37], as well as accuracies (based on the LFs' agreements and conflicts, which are observed in the label matrix) [25]. Snorkel weights the LFs' votes based on these estimations, producing a set of possible labels with corresponding estimated probabilities (*probabilistic labels*).

3. In the last step, a ML classifier is trained using these probabilistic labels. As it can generalize beyond the information that the LFs encode, the classifier is used to improve robustness of the system's predictions, as well as its coverage. As the amount of programmatically labeled data increases, the generalization error of the classifier is said to decrease.

```
from snorkel.labeling import labeling_function

@labeling_function()
def brand_is_steiff(x):
    """
    A toy heuristic expressed as LF:
    Return the label APPROPRIATE if the brand is "steiff"
    Otherwise return ABSTAIN
    :param x:   A data point
    :return:    The label APPROPRIATE or ABSTAIN
    """
    return APPROPRIATE if x.brand == 'steiff' else ABSTAIN
```

Listing 2.1: This Python example shows the definition of an LF that checks whether the brand of a data sample is "steiff". If it is, the function returns APPROPRIATE (a constant that corresponds to the label *appropriate*). If the data sample has a different brand, the function returns "ABSTAIN" (a constant that indicates abstaining).

2.1.1 Data programming application characteristics

A growing body of literature has applied data programming to a specific type of application problem. Among these are tasks from the medical domain such as imaging tasks (Fries et al., 2019 [14]; Saab et al., 2019 [30]), seizure detection in EEG signals (Saab et al., 2020 [31]), improving gene clustering ([11], [12]) and surveillance of medical devices (Callahan et al., 2019 [8]). Other follow-up work has focused on multi-task learning, where LFs may assign labels at various (sub-) task levels ([25], [27], [24]). A previous version of Snorkel [26], called DDLite (Ehrenberg et al., 2016 [13]), was built to create information extraction applications. Altogether, data programming has been applied to a variety of domains and tasks.

In response to a specific task at hand, it is common to extend the general Snorkel pipeline by customized components. An example is the Fonduer framework (Wu et al., 2018 [38]), which specializes in knowledge base construction. A Fonduer application's task is to create a relational database by extracting mentions of relations between entities from input documents. One special feature of this framework is that the human input is three-fold. Next to LFs which assign labels to candidate objects, the users also define the knowledge base schema (capturing different relation types) and Python functions to extract (matchers) or discard (throttlers) candidates before applying LFs to them.

Another example is the work on Snorkel DryBell (Bach et al., 2019 [4]). Snorkel DryBell aims to make the development of ML applications in industry more time- and cost-efficient. In the corresponding study, existing organizational knowledge resources at Google (e.g. internal models, knowledge bases or legacy code) were used as weak supervision sources for content and event classification tasks. To make the translation of organizational knowledge into LFs less redundant, Bach et al. created customized LF templates.
Another system extending Snorkel, Overton, is described in an arXiv preprint from 2019 (Ré et al. [28]). Overton was created at Apple to help engineers in creating, monitoring and improving ML applications in production. An important principle guiding its design is logical independence, i.e. decoupling the management of supervision sources from the ML models. Besides weak supervision sources, the human inputs are descriptions of the input data and the task in a specified schema, as well as definitions of important data slices. Ré et al. stress that the engineers focus on higher-level tasks: They monitor the quality of the application's output and adjust the supervision parts, but are not involved in tuning ML models. These lower-level tasks are automatically carried out by Overton.
In sum, pipeline expansions commonly asked the user for further input, such as specifications of the task or the data structure and data filters. Further, specific pipeline parts have been automated to efficiently use the human intelligence.

Besides various task types, data programming applications are characterized by differences in their users' technical or domain knowledge. Table 2.1 lists users' domain and Python knowledge for a chosen list of applications and shows which interface they used to define LFs. Most applications were targeted at users with domain knowledge who were able to write Python functions. Notably, users in Fonduer's experiment were not familiar with the task domain.
By contrast the Osprey system [7] was specifically targeted at users who are non-coder domain experts. Osprey's authors created customized templates in Microsoft Excel spreadsheets through which users expressed their domain knowledge. According to the domain experts' needs, these templates were prepared for certain types of LFs (e.g. based on keywords/regular expressions). Osprey further comprises LF generators which translate template-based LF definitions into LFs in Python.
Summing up, most applications' users had knowledge of Python. For non-coder users, LF definition was decoupled from their actual Python implementation by providing customized templates.

Approach	Domain knowledge	Python experience	LF Definition Interface
DryBell [4]	Yes	Yes, strong	Python
Overton [28]	Yes	Yes, strong	Python
DDLite [13]	Yes	Yes	Python
Fonduer [38]	No	Yes	Python
Osprey [7]	Yes	No	Customized Excel templates

Table 2.1: Users' domain and Python knowledge, as well as the interface they used for LF definition in different approaches.

Another crucial difference between data programming applications was how they weighted precision versus recall. In most application studies that computed these metrics, no focus on either of the two was explicitly mentioned ([13],[38],[28]). Instead, they reported performance in terms of F1, suggesting an equal importance of both.

Notably, Osprey's creators stressed that their case study's business problem reflected a "precision-oriented workflow", where users only wished to receive preferably less false positives [7]. To the best of our knowledge, no study has published results when applying data programming to a task that was explicitly formulated as a high-recall problem.

2.1.2 Conventional LF definition procedure

In applications relying on data programming, the LFs have a decisive impact on an application's performance. The higher the quality of the LFs, the higher the quality of the probabilistic labels that can be created with them. It is assumed that the parameters of a probabilistic label model can be successfully learned if there are enough LFs and if each LF's performance is better than random (i.e. has an accuracy > 0.5) [3] [36] and if the dataset is balanced.
When writing LFs, users are typically presented with a set of labeled samples, referred to as *development set* since it is used during the development of LFs (e.g. [13], [24]). By inspecting these data samples and their properties, users can get more ideas for their LFs. Though development sets' sizes vary across different experiments, development sets are described as "small", since their size can be considerably lower than that of the training set to label. For instance, different tasks' development sets from [23] included 63 (Crowd dataset), 385 (Radiology dataset) or 2796 (Spouses dataset) candidates.
Usually, LF definition is seen as an iterative process, where users define and adjust several LFs (e.g. [13], [8]). Besides for the inspection of data points, users often also use the development set to evaluate their current LFs' performance (e.g. [10], [13], [35]). Most works did not explicitly mention a time limit that was set for LF development. An exception is the work on Fonduer, where users had a time limit of 30 minutes [38].
Summing up, users usually define LFs in an iterative manner and consult a development set during this process.

2.1.3 Characteristics of user-defined LFs

Some work in which LFs were written by humans and based on domain knowledge has inspected characteristics of these LFs. The most commonly evaluated characteristics are listed below:

- **Accuracy** Individual LFs reportedly have a high accuracy (e.g. [13]). It is believed that there often is a label model misspecification: There might be latent classes in the data (i.e. subsets which users thought of during LF development), on which LFs might make more accurate suggestions than on the whole dataset [34]. Note that it seems to be common practice to compute LFs' accuracy on the data points that they label, i.e. excluding those on which they abstained [1].
- **Coverage** Several studies reported that individual user-written LFs had a low coverage (e.g. [13], [23]). Also, the complete set of LFs rarely reached full coverage [9]. Only a small amount of data points receive a label from more than one LF in the early stages of LF development [23]. In the study on Osprey [7], users also wrote high-coverage LFs. In [14], LFs' coverage was mixed.

[1] https://www.snorkel.org/use-cases/01-spam-tutorial

- **Precision** Individual LFs' precision is typically reported as high (e.g. [23]). In other work, LF precision was described as low (e.g. [14]). This also held for the high-coverage LFs in Osprey [7].
- **Recall** Fries et al. (2019) [14] had LFs with mixed recall.
- **Types and Modalities** Different work has grouped LFs by types (e.g. pattern-based or keyword-based) or data modalities (e.g. text-based or metadata-based [38]). For instance, Ratner et al. (2017) inspected how many LFs per group users wrote [23]. The Fonduer study conducted ablations of LF groups [38]. While such information is insightful (e.g. in designing an end-to-end application for a given task), it is also dependant on the users and the task.

Altogether, most works coincide in the quantitative characteristics they report for LFs that were constructed in this manner. The common observation is that users tend to write LFs with a low coverage and a high accuracy or precision.

2.1.4 Improving the LF creation procedure

Different lines of research have explored ways to optimize the first step in data programming (see Section 2.1), i.e. the definition of LFs. These works either aimed to automatically generate the LFs from user input (Section 2.1.4.1) or to debug user-written LFs using different human-in-the-loop setups (Section 2.1.4.2).

2.1.4.1 Automatic LF generation

Instead of asking users to write LFs, automatic LF generation techniques derive them from other user-provided information. This information can be either actively or passively collected from the users.

According to [27], there is ongoing research on how to use passively collected user information from which LFs could automatically be derived. According to the authors, information to be considered could be eye tracking signals or clickstream data that could be collected during an expert's decision-making process on a data point. Another type of information they mention is created through analyst exhaust. In this setting, analysts (e.g. data scientists) could be asked to explore a dataset by writing code or queries, which could serve as input to an LF generation system.

In 2018, two systems were published that derived LFs from actively collected user data. The BabbleLabble system [16] asks users to annotate a "small" dataset and explain each annotation using natural language. The explanations serve as input to a semantic parser to transform them into candidate LFs. These candidate LFs have to pass a filter bank. Here, semantic filters remove each candidate LF that suggests the wrong label for the annotated example from which it was derived. Also, pragmatic filters eliminate constant, redundant or correlated LFs. From the remaining candidate LFs, the most specific one with the lowest coverage is selected. Carrying out this procedure over all data points, the system creates a set of LFs.
The input to Snuba [36] is more minimalistic, as it requires only a "small" labeled dataset. The Snuba system consists of three components. The first component is a synthesizer that generates candidate LFs from the labeled data. These candidate LFs operate on predefined primitives (e.g. bag-of-words) and they can be expressed through heuristic

models such as decision trees, logistic regressors or clustering techniques. Importantly, the candidate LFs are tweaked so that they abstain on the data points on which they made low-confidence suggestions. The second component is a pruner that aims to select the LF candidate that has the lowest overlap (in terms of covered data points) with the already chosen LFs. Hence, the pruner picks the LF candidate that has the highest weighted average of coverage on still unlabeled data and performance on the labeled data. Snuba's third component, the verifier, is responsible for evaluating when the process of generating and appending LFs should be stopped. This happens if the label model cannot properly estimate LFs' accuracies any more, or when it does not suggest any low-confidence labels any more.

A clear advantage of automatic LF creation techniques is that they are suitable for non-coder experts. If no additional workload should be imposed on users, techniques that rely on passively collected information could be promising. It is important to note that both BabbleLabble and Snuba only create high-quality LFs for the unlabeled data if the labeled dataset is representative. Furthermore, these techniques support an easy integration of information from sources such as knowledge bases.

2.1.4.2 Human-in-the-loop approaches to LF debugging

Human-in-the-loop approaches to LF debugging published in previous studies focused on different shortcomings of user-defined LFs and their implications on data programming applications' performance.

The motivation behind the work by Nashaat et al. [22] was to increase the label model's accuracy by decreasing the number of conflicts between LFs. They combined data programming with active learning: On the basis of the label matrix (see section 2.1, step 2 of the data programming pipeline), they identified data points at which pairs of LFs disagreed. From this pool of data points, they drew some data points using an active learning strategy (uncertainty sampling). They asked users to provide gold labels for these data points. In the label matrix, they subsequently overwrote the LFs' suggestions for these data points with the gold labels. Nashaat et al. observed that their method increased the label model's accuracy, while the degree of this increase depended on the LFs' accuracy. If the LFs' accuracy was "initially low", the improvement gained using their method was comparably low as well. However, they also stressed that they had written the LFs themselves, whereas other approaches had asked domain experts to do so.
In a follow-up work [21], they created WeSAL. In contrast to the method in [22], which focused on conflicts, WeSAL additionally aims at resolving LFs' abstains. Consequently, they also added data points on which LFs abstained to their pool of data points. While using the same datasets as before, they had apparently made alterations of the LFs. Again, they noticed a clear performance decrease, while there was no "significant accuracy boost" if the LFs' initial accuracy had been low.

The Socratic Learning paradigm (Varma et al., published as an arXiv preprint in 2017 [34]) arose from the intuition of a label model misspecification issue: Users might implicitly model subsets of the data during LF development. As a consequence, their LFs might be more accurate on these subsets than on the rest of the data they cover. The Socratic Learning paradigm assumes that the classifier can generalize beyond the

information encoded in the LFs and thereby learn a "better representation" of the training data than the label model. Under this assumption, the disagreement between the two models could point to information that is currently not captured by the label model. In the Socratic Learning paradigm, there is a difference model that selects the features (from the classifier) that have the strongest correlations with the two models' disagreements. Using these features, the label model's accuracy could improve, leading to further improvement in the classifier's decisions. While Socratic Learning aims to automatically update the label model with the features, Varma et al. also conducted a user study. They presented the features in an interpretable way (e.g. by selecting the top n keywords) to the users to guide them in their LF improvements. They found that this "efficient manual debugging" lead to clear error reductions and improvements in F1 score.
These findings were supported in a follow-up paper [35]. In that study, Varma et al. put forward a vision of a framework for training data debugging. In that framework, the whole data programming pipeline and Socratic learning are run once to already obtain an improved label model. Then, the user is presented with interpretable explanations of features selected by Socratic Learning's difference model to inspire LF improvements. In a prototype-based user study, Varma et al. allowed users to improve LFs by adjusting, deleting or writing new LFs. As in their previous user study, they observed a performance boost after the improvements.

The idea of efficiently debugging a set of LFs had already been touched in the work on DDLite [13]. The authors had noticed that once users had created an initial set of LFs, guiding them in their LF improvement was more efficient than trying to inspire improvements via randomly selected data points. They suggested to use intelligent ways of sampling data points to inspire LF improvements. In their work, they applied one intelligent sampling technique, showing users data points that did not receive a label by any LF to inspire increases in coverage. Another sampling technique they suggested was drawing data points on which LFs conflicted the most. Furthermore, they acknowledged that the performance of the classifier trained on the programmatic labels might be more informative than LF statistics to correctly assess the trade-off between accuracy and coverage in LF adjustments.
Cohen-Wang et al. [9] took up these thoughts in an exploratory study. In this study, they examined the effectiveness of different ways to draw training data subsets in guiding LF improvements. In their study, they applied three techniques to draw data points for inspiring LF development.

- Abstain-based strategy: Drawing samples on which the highest number of existing LFs abstained. As there were often data points for which no LF suggested a label, this strategy often drew samples on which all LFs abstained.
- Disagreement-based strategy: Drawing samples on which the highest number of LFs disagreed.
- Random strategy: Randomly drawing samples. This strategy served as a baseline.

They conducted experiments on synthetic (generated binary classification tasks) and real-world data (Amazon reviews) with a simulated human expert and already prepared LFs. For every data point shown, the simulated expert randomly drew one LF out of those prepared LFs that suggested the correct label. They evaluated their results over increasing numbers of queried LFs.
In the experiment with synthetic data, they systematically varied their prepared LFs' properties and assessed the accuracy of the LFs' majority vote. When varying LF

accuracy, they found that the abstain-based and the disagreement-based strategy outperformed the baseline. They also varied the degree to which LFs correlated in which data points they labeled. They found that the performance gain of the abstain-based and the disagreement-based strategy over the baseline increased together with this correlation. For the third property, each LF was generated so that it only suggested one label (i.e. positive or negative), only made correct suggestions and only labeled data points in a certain subset of the data. These data subsets were equally-sized fractions of the unlabeled data, and the same amount of LFs were generated for each of them. Cohen-Wang et al. systematically varied the number of fractions into which the data points (and correspondingly the LFs) were divided. Across different amounts of fractions, they observed that the relationships between the abstain-based and the disagreement-based strategies and the baseline were similar, "but roughly stretched proportionally to the number of partitions".
In the experiment with real data, they automatically created keyword-based LFs consulting a sentiment lexicon. To assess performance, they calculated the coverage of the LFs (i.e. the training set fraction that received a non-abstain label by at least one LF). They also calculated the accuracy of the majority vote of all LFs, the label model trained on the LFs and the classifier in their pipeline. While the abstain-based and the disagreement-based strategies overall outperformed the baseline in terms of accuracy, they did not perceive a difference in coverage.

Summing up, human-in-the-loop approaches LF debugging differed in whether the human was indirectly (i.e. via data labeling) or directly involved in adjusting LFs and the parts in the data programming pipeline that were used to inspire improvements.

3 | Requirements Engineering

When we started our project, the product moderators at bol.com were already using a customized system to detect inappropriate products. We interviewed four members of bol.com's assortment quality team individually. Three of them were product moderators and thereby working with that system on a daily basis. We did not utilize a list of predefined questions but directed the conversation flow into the direction of our interview objectives when needed. The first interview objective was getting to know the concepts and principles governing their system (Section 3.1). The second interview objective was to let them specify the main requirements for a new system (Section 3.2). The third interview objective was understanding where and how their system helped simplify their daily tasks and identifying its limitations (Section 3.3).

3.1 Main principles

The existing system was based on the concept of monitors. We summarize the domain experts' explanations of this concept:
A *monitor* is a single component inside the system that can scan for inappropriate products. Each monitor represents one category of inappropriate products and aims at detecting all products that have to be checked because they might belong to this category.
All items within such a category have a common property. This property is called the *target property* of a monitor. Consider, for instance, the category that contains illegal wildlife products (*exotisch materiaal*). The corresponding monitor for this category should scan for products with the target property that they contain materials such as coral or ivory.
The set of products on which a monitor should be applied is called the monitor's *scope*. Typically, a monitor's scope is a certain subset of all products on the platform. Consider, for example, the category *wegwerpplastic*, which contains all single-use plastic items. Items such as cutlery, plates, straws, and stirrers made from plastic should be inside the scope of a corresponding monitor, so that the monitor's scan could run over them. By contrast, the monitor should not scan over items such as toys made from plastic. These items should be outside of the monitor's scope. An example for a monitor with a large scope is *exotisch materiaal*: Except for a few item types such as books, most products on the platform should be inside the scope of this monitor.
As stated in Chapter 1, each product moderator is specialized in certain categories of inappropriate products. Importantly, the product moderators are responsible for the monitors for each of their specialism categories: They create one monitor for each category, expressing their domain knowledge about the category by means of descriptions

of the target property and the scope of the monitor. Also, they adjust these descriptions if needed to keep the monitor updated. The domain knowledge that is expressed in the product moderator's descriptions of the the target property and the scope is also called the monitor's *logic*.

When running a monitor's scan, all products inside the scope are checked for the target property. They all receive a label indicating whether they seem to be inappropriate (i.e. whether they fulfill the target property) or not. After scanning, the monitor generates a list of potentially inappropriate products (referred to as *candidates* or *flagged products*). In addition to creating and updating the monitor for each of their specialism categories, a product moderator is also responsible for judging whether a candidate suggested by that monitor is actually inappropriate. The process of deciding whether candidates are inappropriate is also called *validation*.
Intentionally, the list of candidates suggested by a monitor should contain products that clearly belong to the category of inappropriate products that the monitor was designed for or unclear cases that need human judgment (e.g. whether a toy gun looks too realistic). What it should not contain are products that are appropriate because they clearly do not display the monitor's target property. In the same vein, products that are outside of the monitor's scope are undesired in this list.

In the system the product moderators were working with when we interviewed them, each monitor's logic a was implemented by one customized *Boolean query*. Such queries evaluate whether or not a specified condition holds for a given product. For instance, they can evaluate the presence of a keyword in the product's description text: They output *True* if the keyword is present and *False* otherwise. They can also check whether a certain product attribute has a certain value (e.g. "material = leather"). Further, these queries can be defined using several conditions that are combined with logical connectors (e.g. "'ivory' in text and price > 100"). In practice, the product moderators often wrote long combinations of different conditions to express a monitor's target property. To express a monitor's scope, they often used negated conditions (e.g. "not 'toy' in text").

3.2 Main requirements

The domain experts named both functional and nonfunctional requirements. The following two subsections present each of them.

3.2.1 Functional requirements

According to the domain experts, the main risk for the system was missing out on products. Having undetected inappropriate products (i.e. false negatives) online constituted a risk for the customer. Consequently, they agreed that the main priority in optimizing the system was recall.
A second risk for the system was generating too many incorrect suggestions. The product moderators stated that they wanted to dedicate as little time as possible to looking at products they do not have to check (i.e. false positives). In consequence, the domain experts agreed that the second priority in optimizing the system was precision.
Altogether, the interviewed domain experts defined product moderation as a high-recall

problem where optimizing precision was desirable, but of lower priority. The domain experts established that the weighting of recall and precision in the F2 score met their needs in a system. Consequently, they decided that the F2 score was the metric to optimize for.

3.2.2 Nonfunctional requirements

The assortment quality team was a team of domain experts. The creation and maintenance of individual monitors would be slowed down if their realization required external help from data scientists or software engineers. Instead, the assortment quality team needed a system enabling them to change a monitor's logic themselves. This requirement entailed that this logic could be customized separately, without having to adjust other components of the monitor. In consequence, all steps necessitated to realize changes in a monitor's logic should be automatized.
The domain experts stressed that product moderation was a fast-paced domain. The longer an inappropriate product was available on the platform, the greater the risk to the customer. Optimally, the product moderators pointed out they would like to identify and eliminate inappropriate assortment as soon as it appeared on the platform. Hence, the system they use should enable them to take action as fast as possible.
As stated in Chapter 1, bol.com's platform policy [6] defined four criteria for any product to fulfill in order to be appropriate for the platform. The assortment quality team already had running monitors for a broad range of categories of items that were inappropriate because they did not fulfill one of these criteria. However, the whole range of possible violations of the criteria was too wide to write monitors for all of them at once. Whenever the assortment quality team found new types of inappropriate items on the platform that did not belong to any of the categories with a running monitor, there was a need for swift action. As a consequence, an important requirement for monitors was that they could be set up fast while achieving an acceptable level of performance. Also, improving a monitor's performance afterward should take as little time as possible.

3.3 Workflow and main limitations

The domain experts described their workflow using the existing system, as well as the limitations and challenges they faced with that system. In doing so, they differentiated between the process of validating candidate products and the process of defining and improving Boolean queries. Below, we summarize their descriptions.

According to the interviewees, it varied across monitors how difficult it was to validate one candidate product. Some monitors' candidates were easy to validate because the target property popped out of the information on the product page. For example, a human could quickly judge whether or not a toy gun was a realistic-looking imitation of a real gun or looked like a toy.
The candidates that were the most difficult to validate were products that needed human judgment, but whose pages did not provide information for this decision in a straightforward manner. They usually appeared in monitors whose target property was implicitly contained, ambiguously described or unaddressed in product pages. An example monitor was *bont* (fur), which intended to flag items made from real fur (including, among others, products in clothing and decoration). For instance,

whether a jacket's fur collar is real or fake may be implicitly described with the phrase '100% polyester'. However, product pages often do not state whether such a material property refers to the whole jacket, to the jacket without the collar or only to the collar. In such cases, product moderators conducted additional research employing information sources outside the platform. Although validating these products was clearly a challenge they faced using their monitoring system, the product moderators explicitly stated that they did not wish to automatize this process. Instead, they wanted any new solution to facilitate the validation by improving how a monitors' logic was implemented.

All interviewed product moderators agreed that the monitors for which Boolean queries were easy to write displayed two main characteristics: First, on the pages of those (and only those) products that fulfilled the target property, this property was explicitly mentioned. Second, the target property was homogenously described over the corpus of products. This meant that there were only a small number of different descriptions used for this property (e.g. in the text on the product page or in the product attributes). As an example monitor, one interviewed expert mentioned *personalization*. This monitor intended to detect products that could be personalized for the customer (e.g. by printing the customer's name on a coffee cup). Since being personalizable was one of the main selling points for such items, this property was explicitly presented on their product pages. The interviewee stated that sellers tended to use the same phrases for this presentation (e.g. asking the customer to mail them after the purchase). Practice had shown that the identified phrases almost exclusively occurred in the contexts of personalization. According to the experts, only a minority of monitors belonged to this group.
The experts further agreed that query writing was hard for the other monitors. Often, product offers inside these monitors' scopes had product attributes that were incorrectly filled in, so that the experts had to rely on keywords. The most difficult part of query writing was finding the right set of keywords (in the text on a product page) that indicated that products were inappropriate. Aiming at high recall, their objective was to include as many inappropriate products as possible, requiring them to find all ways in which these products were described online. However, they found word usage inconsistent across different sellers. To avoid filtering out inappropriate items that were inside the monitor's scope but described using other words than the identified keywords, the product moderators had to conduct a lot of background research on product pages. An initial query typically did not cover enough cases and needed several iterations of refinement (mostly through adding keywords) to achieve acceptable coverage over the corresponding category of inappropriate items.
To include more cases, they had to compare a monitor's suggested candidates to products on the platform and find keywords that could identify additional inappropriate items. This process often caused them to check certain products twice. According to one product moderator, finding a useful keyword was a difficult process. Once a potential keyword was found, it was desirable to ensure that the word did not lead to too many false positives. In theory, this denoted scanning numerous appropriate products' pages for this word. Due to time constraints, this scan often could only be performed in a shallow manner for a larger set of potential keywords.
The product moderators stated that they wanted to overcome the unrealistic problem of having to express a monitor's target property almost perfectly. In consequence, they wished for a solution that could generalize beyond their own descriptions of a monitor's target property.

The useful keywords that the product moderators were able to find in the aforementioned process were often indicative of specific cases only. To prevent an enormous increase in false positives, they usually added sub-conditions for such keywords to the query. One interviewed expert complained about decreased understandability of queries as they got longer and more specific after numerous iterations.
Some categories of inappropriate products were among the specialism categories of more than one product moderator. The effect of decreased query understandability was further strengthened by the fact that not all product moderators were equally proficient in writing Boolean queries. To overcome this obstacle, the most experienced product moderator was responsible for query refinement in such cases. An advantage of this solution were time savings, as this product moderator was faster in debugging. A disadvantage of this solution was that the other team members did not easily understand the reasoning behind all sub-conditions of the query. It thus took them more time to fully grasp a query's functioning. This could be risky when a fast change was needed in several monitors at the same time due to unforeseen events. Therefore, they wished to have a solution in which the implementation of a monitors' logic was easier to understand.

In some monitors, numerous false positives made it difficult to find the cases that product moderators had to judge. Therefore, the domain experts expressed the wish to sort items by their probability of being inappropriate. Such sorting could enable them to manually set a threshold for this probability. Given limited time, they could manually set a probability threshold and process the most severe cases only. Boolean queries, however, do not rank products: They only assign a label to each item. Consequently, a solution that was able to produce a ranking was desired.

3.4 Conclusion

In sum, based on the information above and in correspondence with the domain experts, the main functional requirement for a monitoring system was that it should be optimized according to the F2-score. Furthermore, we identified the following three nonfunctional requirements for such a system:

1. The product moderators formulate a monitor's underlying logic themselves by expressing their domain knowledge via descriptions of the target property and the scope. They can create and adjust a monitor independently of technical experts.
2. The monitors can be quickly created at a reasonable level of performance. This requirement was needed to ensure a fast reaction time to sudden occurrences of new types of inappropriate items on the platform.
3. The monitors can be quickly adjusted to improve performance.

Also, we identified the following three challenges for an improved monitoring system to tackle:

- This monitoring system should generalize beyond their own description of a monitor's target property.
- In this monitoring system, the reasoning behind a monitor's logic should be easy to understand.

- This monitoring system should be able to rank candidate products by their probability of being inappropriate.

4 | Creating Monitors

In our product moderation problem, non-coder domain experts are the users of the desired system. Eventually, we want this system to be directly tailored for their needs. The Osprey system for non-coder domain experts decoupled LF definition based on domain knowledge from the actual translation into Python [7]. A possible solution are customized LF templates for different LF types such as regular expressions or threshold-based heuristics. To design any customized system that meets the users' needs (e.g. by creating templates), it is essential to be familiar with the characteristics of the LFs these users tend to design.
In comparison to the problems for which previous customized systems were build (see Section 2.1.1), our problem stands out by three characteristics.

1. **Our problem is a high-recall problem.** This characteristic resulted from the way the domain experts defined the problem (see Section 3.2.1).

2. **Users rely on their domain knowledge only in their definition of initial LFs for a monitor.** Often, users in a data programming-based application consult a development set to inspire LF definitions (Section 2.1.2). It seems reasonable to assume that the LFs' quality, and consequently the quality of the training labels, depend on the quality of that development set. An underlying concept of the solution demanded by the product moderators at bol.com is that they can create one monitor for each category of inappropriate products. These categories vary in their scope over the products on the e-commerce platform. Finding a flexible way to draw a representative development sets for every category is challenging (and outside of the present work's scope). To cover important, but underrepresented cases, users might for instance need to define data slices (as in e.g. [28]). Such extra steps might slow down the monitor creation process. Furthermore, their daily inspections of the platforms' assortment made our users confident that their domain knowledge, as well as experience with the data, was well-suited to create a monitor. Thus, we did not involve a development set in our monitor creation procedure.

3. **Users need to define the initial LFs for a monitor in a short amount of time.** This characteristic directly follows from the first nonfunctional requirement for a monitoring system (see Section 3.2.2).

Since our problem is set apart from others by these characteristics, it seems reasonable that LFs designed by our users might differ from those written in previous work. As we aim at informing the design of a customized system for the product moderation task, we need to identify the characteristics of these LFs. Furthermore, it can be insightful

to regard their behavior when input into the label model. We are not aware of any previous work that has examined the characteristics of LFs created under the given circumstances or their effect on the label model's performance. To fill this knowledge gap, we ask the following research question:

> **RQ1:** ***What are the characteristics of single LFs that were designed by domain experts in a short amount of time for a high-recall problem and based on their domain knowledge, and what is their effect on the label model's performance?***

Furthermore, we create a prototype of a system that is based on the data programming pipeline. As with the label model, we are not aware of any work that tested the effect of LFs designed under these circumstances on a whole data programming-based pipeline's performance. To fill this knowledge gap and inform further pipeline improvements, we ask the following second research question:

> **RQ2:** ***How does our monitor pipeline perform when a monitor is quickly created by a domain expert based on their domain knowledge for a high-recall problem?***

This chapter is organized as follows: We explain the design of a monitor's underlying pipeline in our prototype (Section 4.1). In Section 4.2, we lay out our experimental methods. In Section 4.3, we report our experimental results. In Section 4.4, we discuss our findings.

4.1 Design of the monitor pipeline

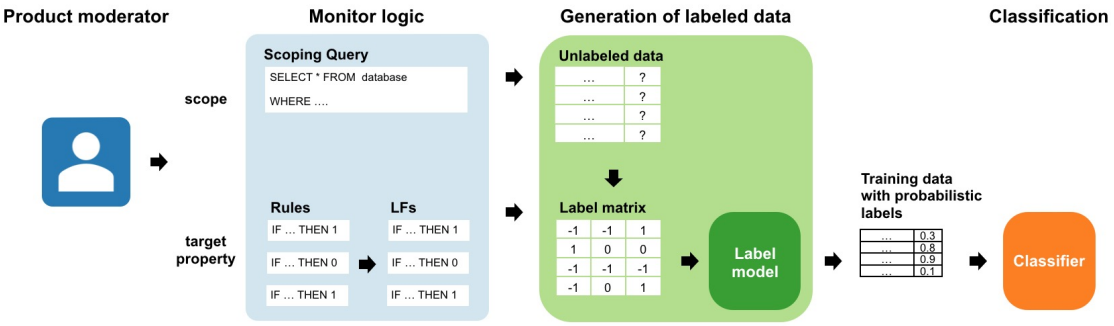


Figure 4.1: The design of the pipeline underlying our monitoring system.

The first nonfunctional requirement for a system for the product moderation task was that the product moderators should be able to create and adjust a monitor's logic autonomously. This requirement was the main driver in the design of a monitor's underlying pipeline. To meet the requirement, we made two design choices.
The first choice was that most components whose adjustment demanded help from a technical expert were kept fixed. Here, fixed meant that their architectures and procedures did not receive any manual modification if the monitor's logic changed (similarly to some pipeline components in [28]). Instead, they were automatically retrained or they just stayed the same. The second design choice was that a monitor's logic could be implemented by tweaking only two adjustable components.
Figure 4.1 illustrates the monitor pipeline: Each monitor implements one category of inappropriate items that arises from bol.com's platform policy [6]. For a category

they are specialized in, the product moderator defines the logic of the corresponding monitor using their domain knowledge: To express the monitor's scope, they write a scoping query that can collect all products through which the monitor should scan from a database. In addition, they write rules to express the monitor's target property. Consequently, the scoping query and the rules are the adjustable components in the pipeline.
Once the domain expert has created or adjusted a monitor, the following steps happen automatically:

1. The scoping query is run, which yields an unlabeled dataset.
2. The rules are translated into LFs. The LFs are applied to the unlabeled dataset, which results in a label matrix (containing each LF's vote for each data sample). This label matrix serves as input for Snorkel's label model. During training, the label model learns estimations of the LFs' accuracies and correlations. The LF's votes for each data sample are weighted according to the label model's estimations and combined to form probabilistic labels. This results in programmatically labeled training data.
3. This step is optional: If a validation set with gold labels (provided by a domain expert) is available, different combinations of feature groups are tested with the same type of classifier on the validation data. The feature combination that lead to the best-performing classifier on the validation samples is, from then on, extracted by the pipeline. If no validation data with gold labels is available, the pipeline extracts a combination of all feature groups.
4. Feature vectors are extracted for all training samples. A classifier is trained using these feature vectors and the probabilistic labels.

The pipeline's classifier will be used to classify the data samples within the monitor's scope. If the probability of the label *inappropriate* for a product is above a certain threshold, this product will be presented to the product moderator as a candidate requiring manual judgement. In principle, this threshold can be set by the product moderator to adjust the number of cases to inspect based on the time they have. For the present work, we decided to fix the threshold at 0.5 to be consistent across monitors.

4.2 Methods

In this section, we describe the methods for our experiments.

4.2.1 Categories

The product moderators selected six categories of inappropriate items for our case study. These categories were called *bont* (fur), *exotisch materiaal* (illegal wildlife products), *magneetballetjes* (magnetic balls), *messen* (knives), *rookwaren* (smoking-drug related items) and *wegwerpplastic* (single-use plastic). Please note that the interpretations of the categories in this research represent simplified versions of those that existed when we created the monitors. Table 4.1 presents an overview of the categories' main characteristics.

- ***bont*** The category *bont* arises from bol.com's principle to refuse items produced at the expense of animals' well-being. *Bont* contains a subset of such products. Inappropriate items in this category are made from animal skin (with hair or fur on it) and real fur of animals such as raccoons, badgers, foxes, minks, coyotes or rabbits. Importantly, leather, animal skin without hair and sheepskin or materials that were produced by shearing or combing animals are allowed. Products made from those materials consequently do not belong to *bont*.
- ***exotisch materiaal*** The category *exotisch materiaal* is governed by a similar principle as *bont*. In contrast to *bont*, which focuses how certain types of materials were produced, *exotisch materiaal* forbids any item containing materials made from endangered animals. The product moderators use the CITES list[1] to look up which animals fall under this category. Example inappropriate products that belong to *exotisch materiaal* are ivory decorations, coral rings and bags made from crocodile leather.
- ***magneetballetjes*** The category *magneetballetjes* is aimed at protecting children. Magnetic building kits are popular toys for children, since they are associated with the support of motor skills and coordination. Despite these potential advantages, small components inside such kits can be dangerous when swallowed by toddlers. Inside the human body, magnetic items may stick together and cause serious injuries. To prevent this, the category *magneetballetjes* comprises magnetic balls or cubes that are small enough to be swallowed. If a product such as a toy kit contains magnetic items with a diameter lower than (or equal to) 4 cm, that product is inappropriate.
- ***messen*** The category *messen* implements article two, paragraph one, subparagraph one and two of the Dutch weapon law Wet, Wapens en Munitie[2]. The items inside the category *messen* are butterfly knives, various types of jack knives, knives with two-sided lemmets and foldable knives that are longer than 28 cm when unfolded.
- ***rookwaren*** The category *rookwaren* contains ingredients and items associated with drugs. Here, the primary focus is on drugs (e.g. cannabis), while the secondary focus is on products that display characteristics of drug-related items (e.g. shishas, vaporizers or herb grinders). The following items belong to the category *rookwaren*: Products that fall under the Dutch Opiumwet[3] like cannabis, products that fall under the Dutch Tabaks- and Rookwarenwet[4] like e-sigarets and products that are typically used for drug-related ends like NO_2 whippets. Excluded from this category are products that are being used for medicinal uses, like CBD-oil.
- ***wegwerpplastic*** The category *wegwerpplastic* originates from the European Union's Strategy for Plastics in Circular Economy[5]. The interpretation of this category is based on a fact sheet the European Union published on single-use plastics[6]. This fact sheet describes ten types of waste items which will have to be removed from

[1] https://www.rvo.nl/onderwerpen/agrarisch-ondernemen/beschermde-planten-dieren-en-natuur/handel-en-vervoer/wel-geen-cites-soort

[2] Wet van 5 juli 1997, houdende regels inzake het vervaardigen, verhandelen, vervoeren, voorhanden hebben, dragen enz. van wapens en munitie, version from 01.10.2019

[3] Wet van 12 mei 1928, tot vaststelling van bepalingen betreffende het opium en andere verdoovende middelen, version from 19.07.2019

[4] Wet van 10 maart 1988, houdende maatregelen ter beperking van het tabaksgebruik, in het bijzonder ter bescherming van de niet-roker, version from 17.11.2018

[5] Communication from the Commission to the European Parliament, the Council, the European Economic and Social Committee and the Committee of the Regions. A European Strategy for Plastics in a Circular Economy SWD(2018) 16 final

[6] https://ec.europa.eu/environment/waste/pdf/single-use_plastics_factsheet.pdf

2021 on. The previous monitor that the product moderators had written for *wegwerpplastic* concentrated on a certain type of items. To stick to this decision, we focused on the same items (cotton buds, cutlery, plates, straws, stirrers and cups) in our research.

Category	Expert	Experience	# Product groups	# Item types
bont	1	thorough	high	high
exotisch materiaal	1	intermediate	high	high
magneetballetjes	2	intermediate	intermediate	very small
messen	2	basic	small	small
rookwaren	3	intermediate	intermediate	intermediate
wegwerpplastic	2	thorough	intermediate	intermediate

Table 4.1: Characteristics of the selected categories: The product moderation expert specialized in the category named by index (Expert), the expert's experience with that category (Experience), the number of different product groups (from the product catalogue) that are inside the scope (# Categories) and the number of distinct types of items that may have the target property (# Item types).

Summing up, the product moderators selected categories containing highly diverse types of inappropriate items to translate into monitors.

4.2.2 Datasets

In the present study, we had four types of datasets for every monitor which we used for different purposes. Table 4.2 provides an overview of these datasets. Below, we describe how these datasets were created.

Dataset	Purpose	Gold labels
training	training the classifier	No
validation	intermediate evaluations during monitor creation and improvement (e.g. computing LF statistics, LF ablation analysis, feature group ablation analysis)	Yes
test	final evaluation after all monitor improvements (Section 5.2.4)	Yes
development	drawing our third (supervised) type of inspiration set to improve monitors in Chapter 5	Yes

Table 4.2: An overview of the four types of datasets we used in this work. For each type of dataset (Dataset), we indicate what its main purpose was (Purpose) and whether gold labels needed to be provided by the product moderators (Gold labels).

For each of the selected categories, we asked the product moderators to describe under which conditions products fall in the scope of a corresponding monitor. Together with the product moderators, we translated these conditions into a query in Google BigQuery[7]. We ran this *scoping query* to obtain all data from the webshop that was inside the scope.
Because of time constraints and the need for labeled data to evaluate our approach, we agreed with the product moderators that they would be able to label up to 1500 products

[7]https://cloud.google.com/bigquery

Dataset		*bont*	*exotisch materiaal*	*magneetballetjes*	*messen*	*rookwaren*	*wegwerpplastic*
train	scope	7371	7369	2288	1264	1011	6794
	historical	262 (262)	57 (57)	28 (28)	2 (2)	60 (60)	570 (570)
	total	**7633 (262)**	**7426 (57)**	**2316 (28)**	**1266 (2)**	**1071 (60)**	**7364 (570)**
dev	scope	362 (25)	303 (0)	335 (0)	210 (17)	162 (0)	359 (29)
	historical	44 (44)	9 (9)	5 (5)	0 (0)	10 (10)	95 (95)
	total	**406 (69)**	**312 (9)**	**340 (5)**	**210 (17)**	**172 (10)**	**454 (124)**
valid	scope	356 (11)	308 (0)	319 (2)	209 (17)	162 (1)	350 (23)
	historical	44 (44)	10 (10)	5 (5)	1 (1)	11 (11)	95 (95)
	total	**400 (55)**	**318 (10)**	**324 (7)**	**210 (18)**	**173 (12)**	**445 (118)**
test	scope	673 (26)	608 (1)	678 (0)	420 (27)	322 (1)	741 (60)
	historical	87 (87)	19 (19)	10 (10)	1 (1)	20 (20)	190 (190)
	total	**760 (113)**	**627 (20)**	**688 (10)**	**421 (28)**	**342 (21)**	**931 (250)**

Table 4.3: Composition of the training (train), development (dev), validation (valid) and test sets for the monitors. The table shows how many products in each set were obtained using the scope query (scope), how many were historical data (historical) and how many the dataset contained in total (total). The numbers in parentheses indicate how many of these products were known to be inappropriate cases.

per monitor. The 1500 products were randomly taken from the scoping query's results. From these labeled samples, we randomly drew 25% as a development, 25% as a validation and 50% as a test set. The remaining unlabeled data obtained by the scoping query was kept as a training set. We uploaded the development, validation and test datasets to a company-internal labeling platform and asked the product moderators to label them. Since this platform could not access all data samples, the sizes of the three datasets decreased during this procedure.

Before the labeling, bol.com's assortment quality team had already successfully detected numerous inappropriate products. The team had also created lists of products that had been recently removed from the web shop in the meantime. For every category, we crawled the data of such products (which we call *historical data*): We used the last time point before a product had been taken offline and crawled the information that had been exposed on the product's page at that time. We split the resulting historical dataset into 60% training, 20% test, 10% validation and 10% development data.

In the last step, we combined the scope data with the historical data. Table 4.3 shows the composition of the training, development, validation and test datasets for each category's monitor.

4.2.3 Monitor creation procedure

In the requirements engineering interviews, the product moderators had stressed their need to quickly set up a monitor with acceptable performance for a given category. To simulate a time restriction, we invited them to a one-hour monitor creation session.

During the monitor creation session, they were asked to translate all their domain knowledge of a target property for a certain category into rules. To maintain consistency across categories, we specified an if-condition-then-outcome structure that each rule should follow. Conditions could be formulated over a set of predefined item properties like the price per piece or the text (product title and description concatenated). We gave examples for certain types of basic conditions. These basic conditions included the presence/absence of keywords, regular expressions and numerical comparisons. The labels assigned to data samples could be either *appropriate* or *inappropriate*. As an example, the toy LF from Listing 2.1 would be translated into *"IF brand_clean = 'steiff' THEN APPROPRIATE"* (see also Figure 5.3). Overall, we allowed nested conditions to account for special cases and avoid major contradictions between rules.
At any point in time, the product moderators were allowed to take a look at the set of rules they had already created and make refinements. However, since they did not write code for their rules, the product moderators were not able to assess rule performance during the session. After the sessions, we transcribed the rules into LFs in Python 3.7[8].

4.2.4 Features

We encoded three different sources of product information in features. The text on a product page had been the focus point of the queries the product moderators had written in their existing system. To include this source of information, we extracted textual features (Section 4.2.4.1) as part of our pipeline.
According to the product moderators, product attributes played a decisive role during the detection of inappropriate items. The product moderators produced a list of all product attributes which helped them detect inappropriate items. Some of the selected product attributes were categorical (e.g. product brand), whereas others were numerical (e.g. product length). Out of these, we computed categorical (Section 4.2.4.2) and numerical attribute features (Section 4.2.4.3), respectively.

4.2.4.1 Textual features

For our textual features, we combined the product title, product description and material description to one text field ("product text"). Directly after crawling the datasets, this product text still had to be cleaned. This was done with the following 4 steps:

1. We cut off undesired text snippets. For instance, we removed artifacts from html tags (e.g. 'br' or 'div'). As manual inspection had shown that there were several words that consisted of three times the same letter only (e.g. 'ccc'), we also removed those.

[8] http://www.python.org

2. We eliminated all links to web pages and ordinal numbers (e.g. '1st', '3rd'), as well as punctuation. We also deleted mentions of measurements (e.g. '3 cm'), since this information was, if important for a monitor, already encoded in the numerical features.

3. We removed Dutch stopwords using the Python package stop-words[9].

4. We lemmatized each word using the Python library spacy[10].

Consequently, after preprocessing, we had product texts consisting of lemmas of all words without stopwords.
Using these texts, we computed two types of textual features. The first textual feature type was a TF-IDF vector. This vector was created in two steps:

1. Using the Python library scikit-learn[11], we computed TF-IDF values for all words that occurred in at least 10 training samples of a monitor, but didn't appear in more than 75% of the training set. These cut-off values were chosen because they had yielded the best results in a proof of concept experiment on *wegwerpplastic* that had carried out at the beginning of the project.

2. To reduce the size of the resulting feature vector, we decided to continue with 1000 of these words. This number of words was chosen since it had also resulted in the best performance in the proof of concept experiment. In selection of these 1000 words, we aimed to maximize the dependence between the words and the probabilistic training labels. The χ^2 statistic is a standard test for independence of two variables. If the p-value is low (usually < 0.05), the null hypothesis (independence between the two variables) can be rejected, suggesting a dependency between the variables. For each word selected in the previous step, we calculated the χ^2 statistic between that word's TF-IDF values across training samples and the probabilistic labels of these training samples. We selected the 1000 words for which the p-evalue was the lowest. Using this procedure, we obtained a textual feature vector that represented the TF-IDF values of 1000 words for each data sample.

The second textual feature type was a fastText vector. As in the TF-IDF computation, we only regarded words that did not appear in more than 75%, but in at least 10 of the training samples in all further steps. We used a pre-trained fastText model for Dutch [15]. This model had been trained on a corpus vocabulary from Common Crawl and Wikipedia. The resulting fastText vector for a given word had length 300. For each product text, we computed the average fastText vector across words. Consequently, we arrived at a fastText feature vectors of length 300 for every data sample.

4.2.4.2 Categorical attribute features

Our categorical features were based on the following attributes (note that the names were adapted by us):

[9] https://pypi.org/project/stop-words/
[10] https://spacy.io/
[11] https://scikit-learn.org

- **brand** The product's brand. The attribute's value can either be a brand name or 'Unknown'. Due to the high number of possible brand names, we only kept the most frequent 25% of brand names for each chunk (see next point). The remaining values were replaced with "other_" followed by the name of the chunk.
- **chunk** One (company-internal) way to group products is to group them by so-called chunks. For example, there are individual chunks for sneakers, cake decorations, guitars and accessories for bread machines. Overall, there are more than 5900 chunks. This product attribute contains the name of the chunk a product belongs to.
- **dishwasherSafe** An attribute indicating whether a product is dishwasher-safe. It can be True, False or Unknown.
- **disposable** An attribute expressing whether products in categories like cutlery were disposable. It can be True, False or Unknown.
- **forChildren** An attribute that indicates whether children are the target users of a product. It can be True, False or Unknown.
- **kitchenGrinderKind** Given that a product is a kitchen grinder, this attribute further specifies the type of kitchen grinder. Its value can either be a grinder type or 'Unknown'.
- **material** The product's material. The attribute's value can either be a material name or 'Unknown'.
- **ovenproof** An attribute that states whether a product is ovenproof. It can be True, False or Unknown.

We one-hot encoded each considered categorical attribute and concatenated the resulting vectors.

4.2.4.3 Numerical attribute features

Some of the numerical attributes we considered were given in different measurement units across products. We standardized such attributes to the smallest of the present measurement units. Further, we divided some existing attributes by the number of pieces that a product contained. This yielded the following attributes:

- **numberOfPieces** If the product was a bundle, this attribute contained the number of items.
- **packageLength** The product package's length, given in millimeters.
- **packageWidth** The product package's width, given in millimeters.
- **price** The product's price, given in euros.
- **pricePerPiece** The product's price in euros divided by numberOfPieces.
- **productLength** The product's length, given in millimeters.
- **productWidth** The product's width, given in millimeters.
- **weightPerPiece** The product's weight in milligrams divided by numberOfPieces.

All aforementioned attributes were either given as numbers or marked as unknown. For every numerical attribute, we first replaced any missing values by the median value. Then, we standardized all values by subtracting the mean and subsequently dividing by the standard deviation. This way, each numerical attribute got represented by a unit variance scaled feature. We subsequently appended all features for a monitor to achieve a numerical attribute feature vector whose length was equal to the number of numerical attributes included.

4.2.5 Classifier

As the main focus of this work were the LFs designed by the domain experts and the overall pipeline for monitors, we did not investigate the usage of complex classifiers. In line with the official Snorkel introduction tutorial[12], we utilized a simple Logistic Regression classifier. We used the same parameter settings as in this tutorial (categorical cross-entropy as loss; Adam optimizer with a learning rate of 0.01).

4.2.6 Evaluation procedure

In this research, general performance was assessed using three metrics. All models (i.e. both label model and classifier) we used assigned a probability for the labels *appropriate* and *inappropriate* to each data sample. For our general performance metrics, we used a threshold on a data sample's probability of the label *inappropriate*: The sample was treated as *inappropriate* if this value was above 0.5 and as *appropriate* otherwise.

- **Recall** The fraction of inappropriate products correctly classified as *inappropriate*, computed with the formula $\frac{TP}{TP+FN}$. Here, TP is the number of true positives (i.e. the inappropriate products that were correctly assigned the label *inappropriate*) and FN is the number of false negatives (i.e. the inappropriate products that were assigned the label *appropriate*).
- **Precision** The fraction of products classified as *inappropriate* that were indeed inappropriate, expressed by the formula $\frac{TP}{TP+FP}$. Here, FP is the number of false positives (i.e. the number of appropriate products that were assigned the label *inappropriate*).
- **F2** A weighted average of precision and recall, defined as $5 * \frac{precision*recall}{4*precision+recall}$. This was the main metric for which we aimed to optimize the monitoring system (see Section 3.2.1).

To answer our research question RQ1, we analyzed different LF properties for every monitor.
We investigated the LFs' coverage as a set by creating a histogram showing which fraction of the training set received a certain number of labels per sample.
We also investigated how confident the label model trained on the LFs' suggestions was that single training data points had the label *inappropriate*. We created a histogram that showed the distribution of predicted probabilities of the label *inappropriate* across training data points.
We also examined properties of individual LFs. One property was the effect that each LF had on the label model's performance. We calculated this effect with an LF ablation analysis on the validation set. Here, effect denotes the label model's difference in performance between using all LFs ($performance_{all}$) and using all LFs but the one whose effect was computed ($performance_{without}$): $effect = performance_{all} - performance_{without}$. If $performance_{all}$ was nonzero, we presented this effect in %. Using this formula, we calculated each LF's effect on the label model's recall, precision and F2 score. In addition, we computed the following statistics for each individual LF with Snorkel's LFAnalysis[13]:

[12]https://github.com/snorkel-team/snorkel-tutorials/blob/master/spam/01_spam_tutorial.ipynb; accessed 08-10-2019

[13]https://snorkel.readthedocs.io/en/v0.9.5/packages/_autosummary/labeling/snorkel.labeling.LFAnalysis.html

- **Polarity** All unique labels (excluding abstains) an LF assigned across training data points
- **Coverage** On what fraction of the training set an LF assigned a label (i.e. not abstain)
- **Overlaps** On what fraction of the training set an LF and at least one other LF assigned a label
- **Conflicts** On what fraction of the training set an LF and at least one other LF assigned different labels
- **Correct** To how many validation data points an LF assigned the correct label
- **Incorrect** To how many validation data points an LF assigned an incorrect label

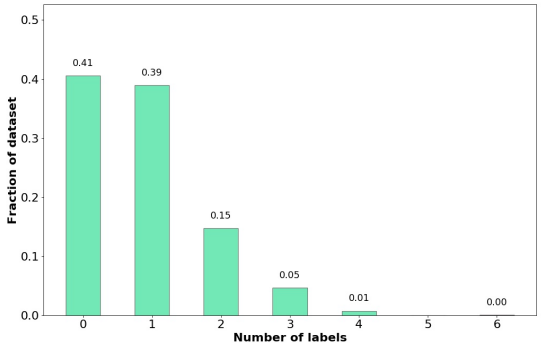
To answer our research question RQ2, we investigated how our pipeline performed on the detection task. We applied the label model trained on the LFs to the training set to receive probabilistic labels for our training samples. We filtered out all training samples that had received the probability 0.5 for both labels. According to [36], such samples could hurt performance since they had not received a label by any LF.
Using this filtered training set, we conducted a feature group ablation analysis. We created all possible combinations of feature groups. The three feature groups we had were categorical features (described in Section 4.2.4.2), numerical features (described in Section 4.2.4.3) and textual features (described in Section 4.2.4.1). Note that the combinations that contained textual features appeared twice, with these features represented by either a TF-IDF or a fastText vector. We did not include both types of textual features in order to not over-represent that information in the feature vectors. We trained one classifier for each feature group combination and computed the performance metrics on the validation set. We then visualized all classifiers' performance (using the label model's performance as a baseline) in a histogram.

4.3 Results

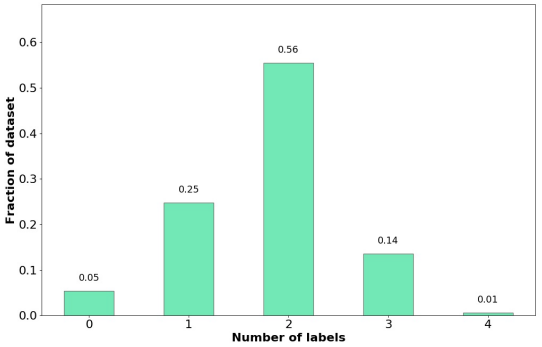
In this section, we present the results of our experiments.

4.3.1 Number of labels per sample

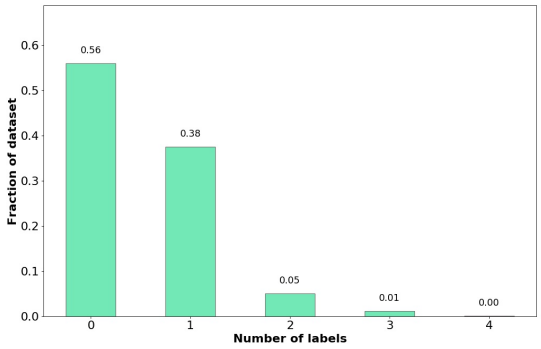
For every monitor, Figure 4.2 shows which fraction of the training set received how many labels per sample. In almost all monitors, a considerable amount of data (31-56%) remained unlabeled. The monitor for *magneetballetjes* stands out as it has the highest number of unlabeled training samples. The exception to this zero-label phenomenon is the monitor for *exotisch materiaal*, where only a small subfraction (around 5%) of the training set received no label.
The monitor for *exotisch materiaal* is also the only one in which the majority of samples were labeled by at least two LFs. In the other five monitors, most of the labeled samples (38-58%) received only one label. The histograms for these monitors display a notable downward trend: The higher the number of labels per sample, the lower the fraction of the training set that received this number of labels. Overall, the step size of this decrease diminishes as the number of labels increases. The biggest difference between dataset fractions can be found when moving from one to two labels per sample. The monitors for *bont*, *magneetballetjes* and *rookwaren* are the only ones among these five in which the fraction of training samples with three labels is still in the single-digit percentage area.



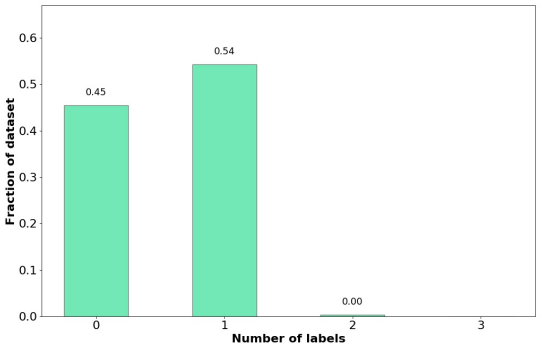
(a) *Bont*



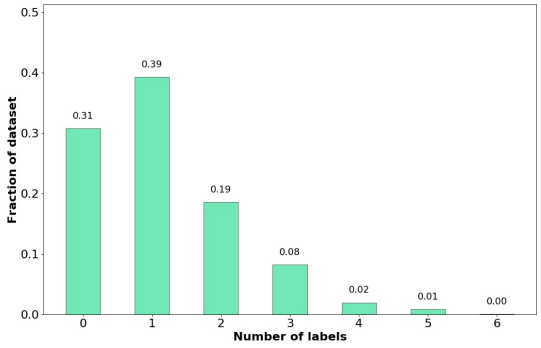
(b) *Exotisch materiaal*



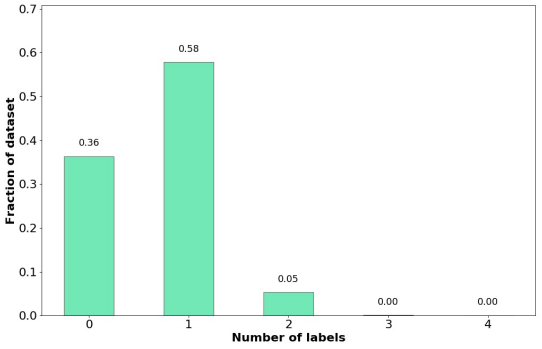
(c) *Magneetballetjes*



(d) *Messen*



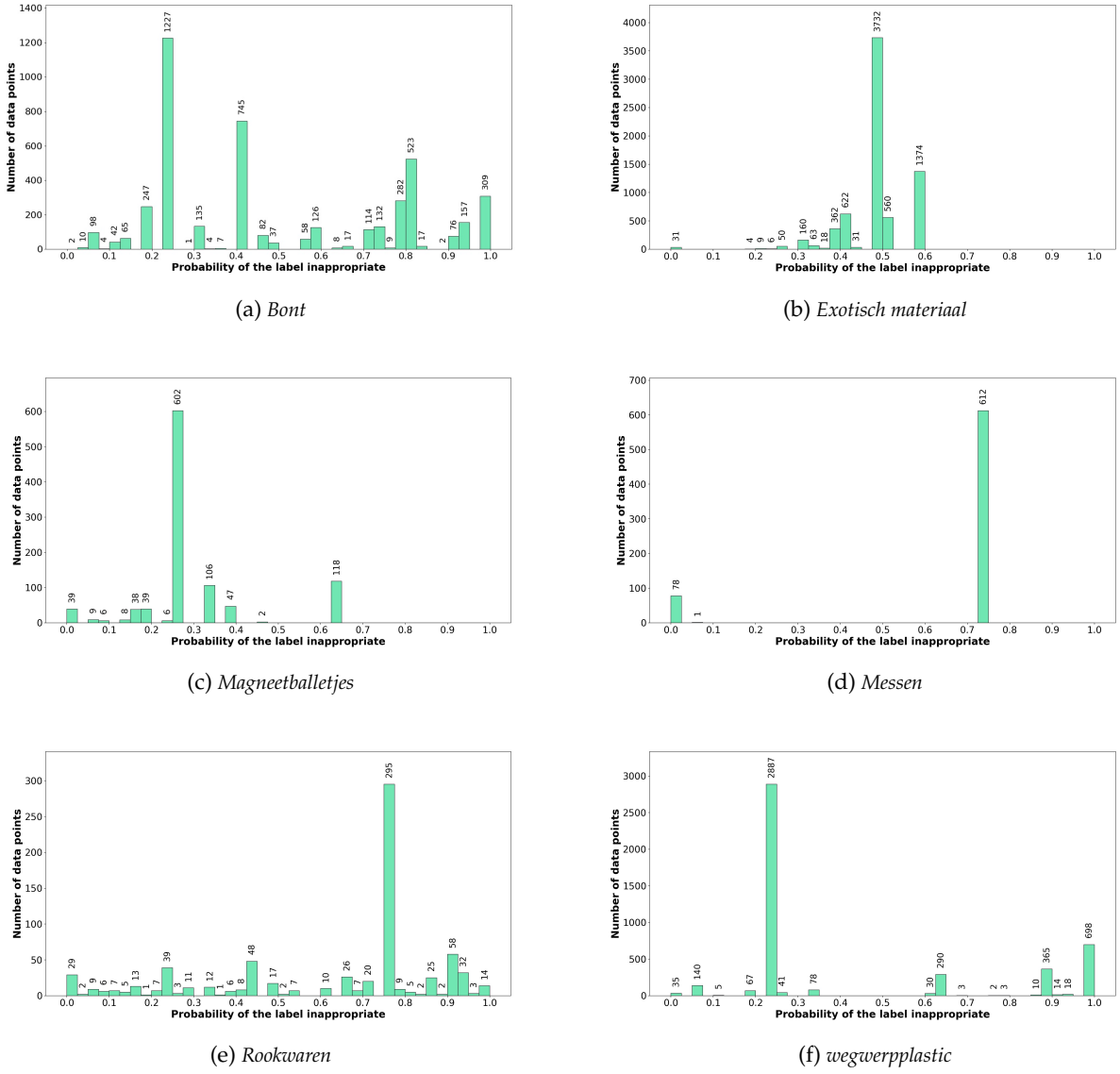
(e) *Rookwaren*



(f) *wegwerpplastic*

Figure 4.2: The fractions of the training sets that received a certain number of labels per sample.

Further differences occur in the maximum number of labels per sample. With up to six labels per sample, this number is the highest in the monitors for *bont* and *rookwaren*. Notably, no training sample for the monitor for *messen* obtained more than 2 labels.



(a) *Bont*

(b) *Exotisch materiaal*

(c) *Magneetballetjes*

(d) *Messen*

(e) *Rookwaren*

(f) *wegwerpplastic*

Figure 4.3: The distribution of probabilities for the label *inappropriate* over the training samples as suggested by the label model trained on the whole set of LFs for each monitor. The probabilities are divided into 40 bins of size 0.025 each.

4.3.2 Probability of the label *inappropriate*

We trained a label model with the initial sets of LFs and examined which probability of the label *inappropriate* it assigned to each data point. For every monitor, Figure 4.3 includes a histogram displaying how the probabilities of the label *inappropriate* were distributed across data points in the monitor's training set.
The monitors *bont*, *rookwaren* and *wegwerpplastic* were the only ones in which the label models assigned probability values at both ends of the possible interval (from values close to 0.00 to values close to 1.00). In the other monitors, the label models assigned probabilities at the low, but not at the high end of the interval (the values did not exceed 0.75).
According to the number of bins with nonzero values, the label models of *bont* and

rookwaren output the highest number of distinct probability values for the label *inappropriate*. The monitor *messen* stood out because the probability values were grouped within only three bins.
In most monitors, there was a tendency towards one end of the probability range. For *messen* and *rookwaren*, most samples received a higher probability of being inappropriate. In *magneetballetjes*, most samples received a lower probability of being inappropriate. The monitors *bont* and *wegwerpplastic* showed a tendency towards low values too, but also had a substantial amount of training data that received high values. The monitor *exotisch materiaal* stood out, since the assigned probability values were mostly clustered around 0.50.

4.3.3 LFs' statistics and effects on the label model

Performance \ Monitor	*bont*	*exotisch materiaal*	*magneetballetjes*	*messen*	*rookwaren*	*wegwerpplastic*
Recall	0.98	0.00	0.57	0.11	0.67	0.60
Precision	0.42	0.00	0.17	0.02	0.10	0.54
F2	0.78	0.00	0.39	0.06	0.31	0.59

Table 4.4: The performance of the label model on the validation set using the complete set of LFs across all monitors.

For each LF, Table 4.5 shows the statistics, as well as the effect on the label model. Recall that single LFs' effect on the label model was computed on the validation set. Table 4.4 shows the performance of the label model on the validation set when all LFs were used to allow for comparisons.
On the level of single LFs, a general tendency of an LF's correct and incorrect suggestions on the validation set could be observed. LFs with polarity 0 (i.e. only suggesting the label *appropriate*) made considerably more correct than incorrect suggestions. A lot of LFs that suggested the label *inappropriate*, by contrast, made remarkably many incorrect suggestions. Manual inspection revealed that LFs outputting the label *inappropriate* belonged to three groups, based on their precision and recall on the validation set:

- **High precision and accuracy, low recall** Among others, this group counted most LFs from *wegwerpplastic* and three LFs from *bont*.
- **Low precision and accuracy, low recall** Overall, most LFs belonged to this group. For instance, it contained most LFs from *rookwaren*, all LFs from *messen* and three LFs from *bont*.
- **Low precision and accuracy, high recall** Among others, this group comprised most LFs from *exotisch materiaal* and the LF from *magneetballetjes*.

Within the three groups, the LFs' effects on the label models varied across monitors.

Index	Polarity	Coverage	Overlaps	Conflicts	Correct	Incorrect	%Effect on LM Recall	%Effect on LM Precision	%Effect on LM F2
0	[0]	0.25	0.09	0.04	77	1	0.00	3.03	1.14
1	[0]	0.01	0.01	0.01	3	0	0.00	-1.59	-0.58
2	/	0.00	0.00	0.00	0	0	0.00	0.00	0.00
3	[0]	0.02	0.01	0.00	6	0	0.00	0.00	0.00
4	[0]	0.03	0.02	0.01	9	3	0.00	0.00	0.00
5	[0]	0.01	0.00	0.00	3	0	0.00	0.00	0.00
6	[0]	0.21	0.11	0.07	79	5	0.00	3.03	1.14
7	[0, 1]	0.02	0.02	0.01	2	5	3.70	-1.03	2.01
8	[1]	0.00	0.00	0.00	0	0	0.00	0.00	0.00
9	[1]	0.13	0.10	0.06	11	34	5.56	-27.25	-4.34
10	[1]	0.02	0.02	0.00	5	1	0.00	-0.79	-0.29
11	[1]	0.00	0.00	0.00	4	0	0.00	0.00	0.00
12	[1]	0.00	0.00	0.00	4	0	0.00	0.00	0.00
13	[0, 1]	0.16	0.10	0.06	45	49	70.37	25.64	61.95
14	/	0.00	0.00	0.00	0	0	0.00	0.00	0.00

(a) Bont

Index	Polarity	Coverage	Overlaps	Conflicts	Correct	Incorrect	Effect on LM Recall	Effect on LM Precision	Effect on LM F2
0	[0, 1]	0.72	0.66	0.65	10	204	-0.50	-0.02	-0.09
1	[1]	0.00	0.00	0.00	2	0	0.00	0.00	0.00
2	[1]	0.10	0.10	0.10	7	32	-1.00	-0.04	-0.19
3	[0]	0.02	0.02	0.02	10	0	-1.00	-0.05	-0.21
4	[0]	0.00	0.00	0.00	0	0	-1.00	-0.05	-0.20
5	[0]	0.83	0.66	0.65	242	0	-1.00	-0.04	-0.19
6	[0]	0.11	0.10	0.08	39	2	-1.00	-0.05	-0.19

(b) Exotisch materiaal

Index	Polarity	Coverage	Overlaps	Conflicts	Correct	Incorrect	%Effect on LM Recall	%Effect on LM Precision	%Effect on LM F2
0	/	0.00	0.00	0.00	0	0	0.00	0.00	0.00
1	[0]	0.04	0.02	0.00	9	0	0.00	0.00	0.00
2	[0]	0.31	0.06	0.03	114	0	0.00	32.35	17.74
3	[1]	0.08	0.03	0.03	4	30	100.00	100.00	100.00
4	[0]	0.07	0.02	0.00	20	0	0.00	0.00	0.00
5	[0]	0.02	0.02	0.01	6	0	0.00	0.00	0.00

(c) Magneetballetjes

Index	Polarity	Coverage	Overlaps	Conflicts	Correct	Incorrect	%Effect on LM Recall	%Effect on LM Precision	%Effect on LM F2
0	/	0.00	0.00	0.00	0	0	0.00	0.00	0.00
1	[1]	0.49	0.00	0.00	2	96	100.00	100.00	100.00
2	[1]	0.00	0.00	0.00	0	0	0.00	0.00	0.00
3	/	0.00	0.00	0.00	0	0	0.00	0.00	0.00
4	[0]	0.04	0.00	0.00	11	0	0.00	2.04	1.18
5	[0]	0.02	0.00	0.00	4	0	0.00	0.00	0.00

(d) Messen

Index	Polarity	Coverage	Overlaps	Conflicts	Correct	Incorrect	%Effect on LM Recall	%Effect on LM Precision	%Effect on LM F2
0	[1]	0.42	0.19	0.12	9	65	0.00	-51.85	-27.45
1	[1]	0.05	0.05	0.04	1	7	0.00	0.00	0.00
2	[1]	0.11	0.07	0.04	6	13	0.00	0.00	0.00
3	[0]	0.05	0.05	0.04	5	2	-25.00	-12.64	-16.91
4	[0]	0.09	0.06	0.05	15	0	0.00	6.82	4.41
5	[0]	0.04	0.03	0.03	8	1	0.00	2.38	1.52
6	[0]	0.02	0.02	0.02	6	0	0.00	1.20	0.76
7	[0]	0.04	0.04	0.04	3	0	0.00	3.53	2.26
8	[0]	0.03	0.02	0.02	2	1	0.00	0.00	0.00
9	[0]	0.00	0.00	0.00	3	0	0.00	0.00	0.00
10	[0]	0.08	0.07	0.07	16	0	0.00	4.65	2.99
11	[0]	0.00	0.00	0.00	1	0	0.00	0.00	0.00
12	[0]	0.01	0.01	0.01	1	0	0.00	0.00	0.00
13	[1]	0.01	0.01	0.01	2	3	-12.50	-11.14	-11.64
14	[1]	0.01	0.01	0.00	1	1	0.00	0.00	0.00
15	[1]	0.17	0.13	0.07	4	33	12.50	-1.06	4.41

(e) Rookwaren

Index	Polarity	Coverage	Overlaps	Conflicts	Correct	Incorrect	%Effect on LM Recall	%Effect on LM Precision	%Effect on LM F2
0	[1]	0.02	0.01	0.00	17	1	8.45	3.32	7.38
1	[0]	0.06	0.01	0.00	30	2	2.82	-25.77	-2.26
2	/	0.00	0.00	0.00	0	0	0.00	0.00	0.00
3	/	0.00	0.00	0.00	0	0	0.00	0.00	0.00
4	/	0.00	0.00	0.00	0	0	0.00	0.00	0.00
5	[0]	0.43	0.04	0.00	152	1	8.45	-54.93	-0.54
6	[0]	0.04	0.03	0.00	20	0	0.00	2.94	0.66
7	[0]	0.01	0.00	0.00	5	0	0.00	-1.54	-0.33
8	[0]	0.01	0.01	0.00	0	0	0.00	0.00	0.00
9	[0]	0.01	0.00	0.00	1	1	1.41	0.66	1.24
10	[1]	0.01	0.00	0.00	2	4	0.00	-2.33	-0.50
11	[0]	0.06	0.02	0.00	21	4	4.23	-11.88	1.11
12	[0]	0.00	0.00	0.00	0	0	0.00	0.00	0.00
13	[1]	0.07	0.01	0.00	58	9	66.20	42.05	62.81

(f) Wegwerpplastic

Table 4.5: For each LF in each monitor, this table shows the statistics on the training set (polarity, coverage overlaps, conflicts), the number of incorrect and correct suggestions on the validation set, and the effect on the label model's (LM) performance. Gray lines indicate LFs with zero coverage.

Another apparent difference between individual LFs was their coverage on the training set. In almost all monitors (except for *bont*), there was at least one LF that covered more than 30% of the training set. In *magneetballetjes* and *wegwerpplastic*, these high-coverage LFs output the label *appropriate*. In *exotisch materiaal*, *messen* and *rookwaren*, there were high-coverage LFs suggesting the label *inappropriate*. In *exotisch materiaal* and *rookwaren*, there were also intermediate-coverage LFs that covered between 10.00 and 30.00% of the training data. Most of them suggested the label *inappropriate*. In addition to a few LFs with high or intermediate coverage, all monitors contained numerous low-coverage LFs that covered less than 10% of the training sets and had different polarities. In all monitors except for *rookwaren* and *exotisch materiaal*, there were zero-coverage LFs that did not output any label for any training or validation sample.
In *exotisch materiaal*, the LFs overlapped with others on most of the training samples the covered and had conflicts in most of their overlaps. By contrast, most LFs in *messen* and *wegwerpplastic* overlapped or conflicted with each other on less than 1.00% of the training data.

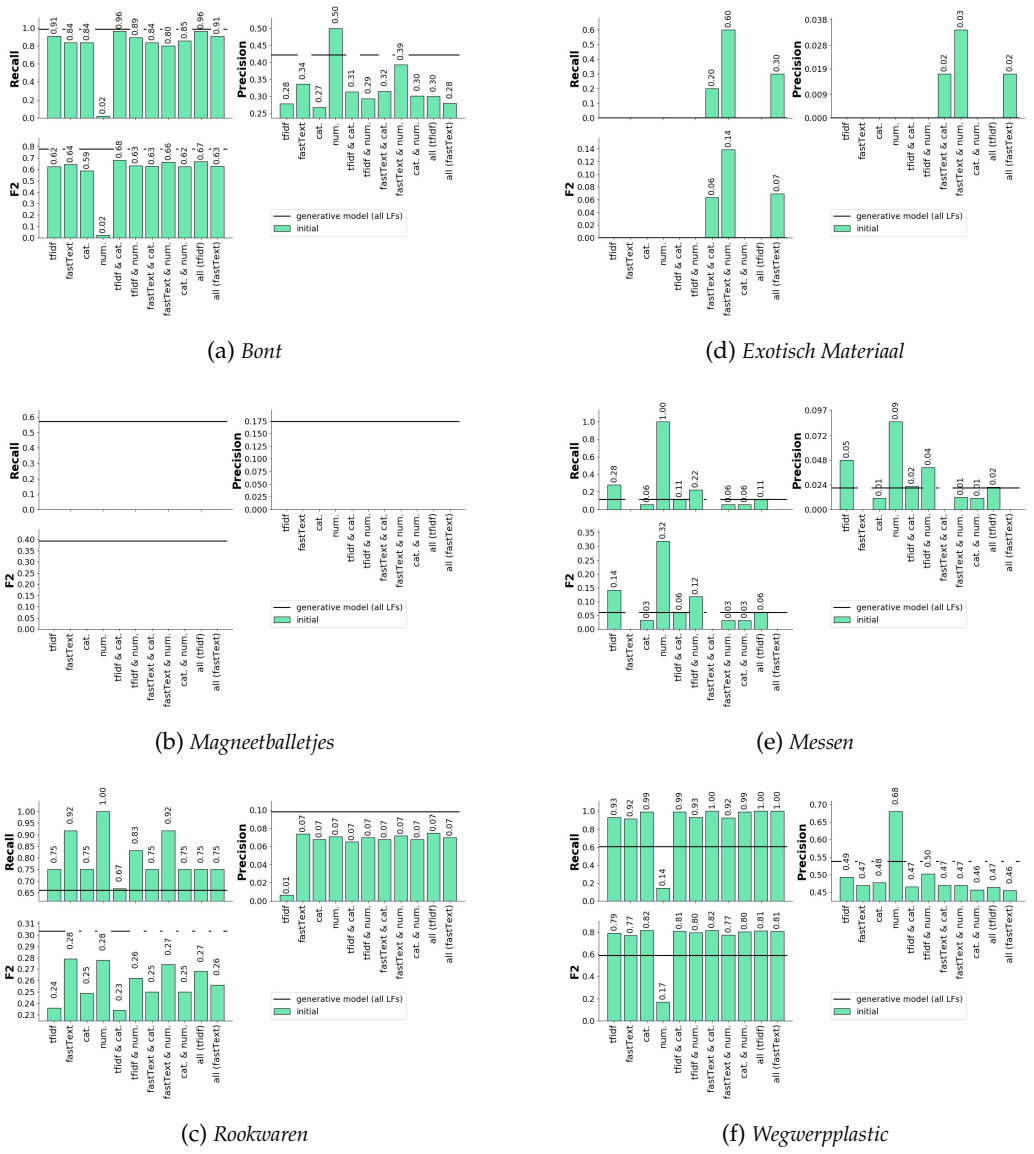
With respect to the single LFs' effects on the label models, we can observe the following patterns:

- Across all monitors except *wegwerpplastic*, there were several LFs among those that did not have zero-coverage whose inclusion had no perceivable (i.e. less than 1.00%) effect on the label model's performance.
- For most other LFs (except for those in *exotisch materiaal*), the inclusion had only a small (less than 5.00%, positive or negative) effect on the F2 score.
- In all monitors except *exotisch materiaal*, we see that those LFs whose inclusion had the strongest impact on the label model output the label *inappropriate*. This effect was clearest in *magneetballetjes* and *messen*, where removing the only LF that suggested that label and covered more than 1.00% of the training data lead to a performance difference of 100.00% (zero performance). *Rookwaren* was the only monitor in which the strongest effects were negative.
- Surprisingly, all LFs but one from *exotisch materiaal* had a strong negative effect on the label model's performance. Excluding one of them lead to a rise in recall from 0.00 to 1.00 (or 0.50 for LF 0).
- Astonishingly, there were LFs outputting only *appropriate*, but still affecting the recall score (LF 3 from *rookwaren* and LFs 1, 5, 9 and 11 from *wegwerpplastic*). Also, LF 13 from *rookwaren* only suggested the label *inappropriate* and made two correct suggestions on the validation data, but had a negative effect on the recall.

An obvious aspect the monitors differed in is their performance when using all LFs. In most monitors, precision of this label model was considerably lower than recall. The highest F2 scores were obtained in the monitor *bont*, followed by *wegwerpplastic*. Notably, the label model from *bont* almost detected all inappropriate products. The monitor *exotisch materiaal* stood out, since its label model's performance with all LFs had been zero on all scores.

4.3.4 Feature group ablation analysis

Figure 4.4 shows the results of the feature group ablation analysis for each monitor. The monitors differed in the scores that their best-performing classifier reached. The highest F2-score was obtained in the monitor *wegwerpplastic* (0.82), followed by *bont*



(a) *Bont* (d) *Exotisch Materiaal*

(b) *Magneetballetjes* (e) *Messen*

(c) *Rookwaren* (f) *Wegwerpplastic*

Figure 4.4: Precision, recall and F2-score results of the feature group ablation analysis. The colors correspond to the feature group combinations used in the discriminative models. The black line indicates the performance of the label model.

(0.68). The other monitors' best classifiers had only low F2 values (0.32 in *messen*, 0.28 in *rookwaren* and 0.14 in *exotisch materiaal*). Notably, all classifier variants in the monitor *magneetballetjes* had zero values in all performance measures.

In all other monitors, the best-performing classifier's precision was clearly lower than its recall and even below 0.1 in *exotisch materiaal*, *messen* and *rookwaren*. *Exotisch materiaal* and *messen* were the only monitors in which this precision was higher than that of the label model. Among these five monitors, *bont* was the only one in which the classifier did not reach a higher recall than the label model.

In only half of the monitors (*exotisch materiaal* , *messen* and *wegwerpplastic*), the classifier outperformed the label model. In *exotisch materiaal* and *wegwerpplastic*, this classifier used a combination of fastText and either numerical (*exotisch materiaal*) or categorical (*wegwerpplastic*) features. In *messen*, the best-performing model relied on numerical

features only. In all other three monitors, the label model outperformed all classifiers.

In addition, the figures display various types of characteristics and differences of the classifiers in each monitor. In the monitor *bont*, all classifiers except for one had very high recall scores, but low precision. In contrast to them, the classifier using only numerical features had by far the lowest recall, but a precision that outperformed all of the monitor's models. As for textual features, the feature group combinations with TF-IDF yielded better results than those with fastText. In the monitor *exotisch materiaal*, the only models that obtained nonzero values were those trained on combinations of fastText features with at least one other feature type. Across those three models, the model with fastText and numerical features scored the highest in all three performance measures. In the monitor *messen*, the models that had the highest recall also got the highest precision scores. Notably, the difference in recall between the best (numerical features) and the second best model (TF-IDF features) was larger than 0.7. The classifiers relying on fastText, fastText with categorical features and fastText with categorical and numerical features had zero performance values. In the monitor *rookwaren*, the models relying on fastText features or numerical features reached the best score. In the monitor *wegwerpplastic*, all classifiers except for one had high recall, but low precision. The other classifier had the highest precision, but the lowest recall. This tendency closely resembles that observed in the monitor *bont*.

4.4 Discussion

In the following lines, we will discuss our results with respect to our first two research questions.

> **RQ1:** ***What are the characteristics of single LFs that were designed by domain experts in a short amount of time for a high-recall problem and based on their domain knowledge, and what is their effect on the label model's performance?***

Our domain experts wrote a small number of high- or intermediate-coverage LFs and numerous low-coverage LFs. This suggests that the domain experts described the monitors' target properties using a few main indicators and then further included small LFs to detect more specific cases. As a set, the LFs had a low coverage, leaving a high number of data points unlabeled. This observation is in line with findings from different data programming application settings [9]. As already recognized by [23], only a few data points received more than one label at this early point of LF development.
LFs that identified appropriate products were very accurate across the data points that they covered. Thereby, they display the same properties as LFs in previous work [13]. In contrast, only a subset of those LFs that detected inappropriate products had a high precision and accuracy. The majority LFs aimed at detecting inappropriate products were characterized by a low precision and accuracy. Among them were some high-recall LFs.
Based on their experience with Snorkel users, Ratner et al. (2017) [23] observed that defining LFs for negative cases could be counter-intuitive, mirrored by low coverage over these cases. Our results suggest that defining LFs with a high coverage based on the domain knowledge in a short amount of time seems to be challenging. Under these settings, users also seemed to have difficulties to define highly-accurate LFs for positive cases. Overall, this problem seemed most prominent in categories with which the

domain experts were less experienced. It could be caused by the product moderators' desire not to miss out on any inappropriate products. It also seems reasonable that they had specific subsets of their data in mind during LF definition (as suggested by [34]). This idea could also explain why domain experts wrote LFs with zero coverage: They might have thought of specific cases that might not have been included by the data crawled with their scoping query.

Overall, individual LFs' effects on the label model seemed to be small. We observed that the greatest positive effects on the performance were caused by LFs detecting inappropriate products. At first sight, this observation appears straightforward: Since precision, recall and F2 concentrate on the positive class and what is suggested to be in that class, it seems reasonable to assume that removing an LF that detects positive cases at least leads to a reduction in recall and F2. Yet, in the monitor *rookwaren*, we found an LF detecting inappropriate products and an LF labeling appropriate products that both had strong negative effects on the recall. This shows that LFs' interplay (i.e. correlations and conflicts with each other) plays a decisive role in their individual effects on the label model's performance. This idea was supported by a special case, *exotisch materiaal*. We observed that domain experts could, under the given circumstances, define LFs that were so contradictory that they strongly impaired the label model's performance. Except for this monitor, the domain experts' experience with a category seemed to be reflected in the label model's behaviour. A small experience appeared to coincide with a small variety of probabilities for the positive class suggested by the label model and weaker label model performance. The category with which the product moderators were most experienced had the highest label model performance.

> **RQ2:** ***How does our monitor pipeline perform when a monitor is quickly created by a domain expert based on their domain knowledge for a high-recall task?***

In general, the whole monitor pipeline produced a classifier that had a promising performance in two monitors when set up under these circumstances. In discussions with the domain experts after our experiments, they confirmed that this performance level was acceptable for an initial monitor version. In the other monitors, the classifier's performance was poor. The classifiers' performance further demonstrated that precision was the monitors' major weakness. This was in line with observations on LFs that detected inappropriate products. The difference in feature combinations that lead to the best-performing classifier in the monitors seemed to justify the automatic feature group ablation step in our pipeline.
According to the common opinion, the main advantage of including the classifier in the data programming pipeline is that it might generalize and cover cases that were not covered by the LFs [23]. In four out of six monitors, classifiers indeed reached a higher recall than label models. It is also assumed that the classifier learns a better representation of classes than the label model [34]. In only two of our monitors, the best-performing classifier (i.e. the one the pipeline used in the end) had a higher precision than the label model, but that precision was still poor. Also, in half of our monitors, the label models' performance was generally better than that of any classifier.
We manually compared the probabilities for the label *inappropriate* predicted by the classifiers and the label models to investigate such surprising results. Though the *magneetballetjes*' label model had reached the third "best" performance across all monitors (on the validation set), all its classifiers had zero performance scores. Our inspection revealed that all classifiers' maximum of suggested probabilities was below that of the label model (0.64). A possible reason for this is the proportion between the number

of categories inside the defined scope and the number of item types with the target property (see Table 4.1). Product text could be written differently across categories and attributes could be filled in differently as well, making learning harder for a classifier. According to the domain experts, products assigned to the wrong product groups in the product catalog are common problem in such categories.
The label model in *exotisch materiaal* had never correctly suggested the label *inappropriate*, resulting in zero performance. Yet, the classifiers managed to correctly assign this label. Our inspection showed that the label model's suggestions for inappropriate products had been slightly below the decision boundary, while the best classifiers' had been slightly above.
In other cases where the label model outperformed the classifiers (e.g. in *bont*), the classifiers' suggested probabilities were usually lower. A general reason for this phenomenon might be missing values or values that are incorrectly filled in. According to the domain experts, these are commonly known issues with products on the platform, since different sellers might have different habits of presenting their products.

5 | Improving Monitors

When creating monitors for six categories of inappropriate products in our first experiment (see Chapter 4), performance was considerably poor in four of them. In this chapter, we address the third nonfunctional requirement we identified for a suitable system. This requirement is that monitors' performance can be improved through quick adjustments (see Section 3.2.2). In the setup of our previous experiment, the product moderators injected their domain knowledge into our pipeline in two ways: According to the pipeline design (see Section 4.1), they defined the monitors' scope through a scoping query and the monitors' target property through LFs.
In our second experiment, we aimed to systematically evaluate the effect of adjusting one type of injected domain knowledge. If a monitor's scoping query is adjusted, the monitor's datasets change. A reasonable consequence would be that the distribution of item types within the datasets changed as well, demanding adjustments of the LFs. In consequence, we assumed that solely adjusting the scoping query without touching LFs would not be reasonable, and that adjusting LFs would be the most generalizable way to improve monitors.

Previous work on improving the LFs or the manner of creating them considered automatic LF generation and human-in-the-loop approaches to LF debugging (see Section 2.1.4). Our first main requirement for a suitable system was that the product moderators formulated a monitor's underlying logic themselves. To create a monitor, they had to explicitly express their domain knowledge when designing LFs. Monitor improvements should follow the same principle. Consequently, automatic techniques in which users can only implicitly inject their knowledge through passively collected information or labeled data points did not appear suitable for our task. Instead, we decided to use a human-in-the-loop approach.
One type of human-in-the-loop approach had been to resolve LF conflicts by asking the user for gold labels on certain data points ([21], [22]). Again, here, the user would not directly adjust LFs, rendering such techniques less suitable for our problem. In Section 2.1.4.2, we reviewed two other lines of techniques in which users were inspired to actively debug LFs: Socratic learning ([34], [35]) and intelligently drawn data points. Socratic learning uses the disagreement between the label model and the classifier as an inspiration for improvements. In this paradigm, it is assumed that the classifier learns a better representation of the classes than the label model [34]. However, in our experimental results, our simple classifier was often outperformed by the label model. We therefore resolved on not employing Socratic learning in the present work.
Instead, we focused on intelligently drawn data points. Previous approaches took LFs' abstains ([9], [13]) or disagreements [9] into account when drawing data points to inspire LF improvements. Also, the classifier's performance has been suggested to provide information to guide LF improvements ([9], [13]). In the present work, we base

one strategy of drawing data points on each of these three indicators. We refer to sets of intelligently drawn data points as *inspiration sets*, since their goal is to inspire LF improvements.
The following are the main aspects that distinguish our experimental setup from that employed in [9]:

- We introduced a new strategy for drawing an inspiration set that was based on the classifier's performance.
- We used the inspiration sets to improve an existing set of LFs that had been created based on domain knowledge only (see Chapter 4). In [9], the same strategy was used to iteratively update a set of LFs over a predefined number of iterations.
- We conducted a study on a real-world task with real (not simulated) users that wrote or adjusted the LFs. [9] suggested this as a direction of future work.
- Our users could not only add new LFs, but also adjust or delete existing ones.
- The task we tackled was a high-recall problem.
- We investigated the usefulness of inspiration sets in inspiring LF improvements in a short amount of time. This time limit was necessary to fulfill our third nonfunctional requirement for a monitoring system (see Section 3.2.2).
- All data points in one of our inspiration sets were drawn at the same point of time. In [9], the changes in LFs were used to constantly update a pool of data points from which the next sample to show to the user was drawn. Our users were non-coder domain experts. Since we aimed to investigate LF improvements in a short amount of time, we avoided introducing time delays by implementing their LF improvements during our experiments. This prevented us from updating the set of data points that could be shown to the user.
- We showed the user the whole inspiration set at once, not single data points. This was another knowledge gap pointed out by [9].

With regard to these differences, we considered three types of inspiration sets in our study. Two of them were *unsupervised*, since they could be extracted without requiring gold labels:

- **Previously unlabeled data** We used an abstain-based strategy to draw this inspiration set. Data points that received no label by any of the initial LFs in a monitor were taken into account.
- **Data on which LFs previously disagreed** We used a disagreement-based strategy to draw this inspiration set. [9]'s disagreement-based strategy drew data points with the maximum number of LF disagreements. Manual inspection of our results from Chapter 4 had shown that a small number of LF constellations tended to produce the maximum number of disagreements per LF. Our inspiration sets were drawn at one time point without considering updates to the set of LFs. To allow for more variety in the LF disagreements represented in the inspiration set, we chose to take all data points on which LFs disagreed into account.

The third inspiration set type was *supervised* (i.e. required gold labels to be extracted):

- **Data most mistaken by the classifier** We used a simple classifier-based strategy to draw this inspiration set. Here, our aim was to allow for correcting mistakes made by the classifier via the LFs. Consequently, we took the data points on which the classifier was most mistaken in its suggestion into account.

Previous work in [9] had systematically varied dataset properties and compared the effects of LF adjustments on the majority vote of LFs, the label model and the classifier. In our experiments on creating monitors (see Chapter 4), LFs' correlations might have influenced the LFs' effects on the label model's performance. If an LF's adjustment was inspired by an inspiration set, its correlations with other LFs might alter. It seems reasonable that such adjustments might affect the LFs' effect on the label model. Research is needed to gain a deeper understanding of how adjustments inspired in an experimental setup like ours can alter LFs' effects on the label model's performance. To fill this knowledge gap, we seek to answer the following three research questions:

> **RQ3a:** ***How do LF adjustments inspired by fast inspection of previously unlabeled data change the LFs' effect on the label models' performance?***
>
> **RQ3b:** ***How do LF adjustments inspired by fast inspection of data on which LFs previously disagreed change the LFs' effect on the label models' performance?***
>
> **RQ3c:** ***How do LF adjustments inspired by fast inspection of data most mistaken by the classifier change the LFs' effect on the label models' performance?***

Furthermore, to inform a practical solution for our third nonfunctional requirement, we seek to answer the following research question:

> **RQ4:** ***Which of our three inspiration set types is best-suited for improving the performance of our whole pipeline in a short amount of time?***

This chapter is organized as follows: We describe our experimental methods in Section 5.1. In Section 5.2, we report the results of our experiments. In Section 5.3, we discuss our findings with respect to our research questions.

5.1 Methods

In this section, we describe our methods.

5.1.1 Drawing inspiration sets

The inspiration sets were all drawn according to different strategies from a monitor's training (unsupervised inspiration sets) or development set (supervised inspiration set). Figure 5.1 provides an overview of our monitor pipeline and shows at which steps the inspiration sets were drawn. Note that we set 100 data points as the maximum inspiration set size. Table 5.1 displays the sizes of all inspiration sets for each monitor.

- **Previously unlabeled data** We selected all data samples from a monitor's training set on which all initial LFs abstained. If this selection yielded less than 100 data points, we kept all of them. Otherwise, we randomly drew 100 data points.
- **Data on which LFs previously disagreed** We selected all data samples from a monitor's training set on which at least two initial LFs disagreed. Again, if the resulting dataset comprised more than 100 data points, we randomly drew 100 data points.

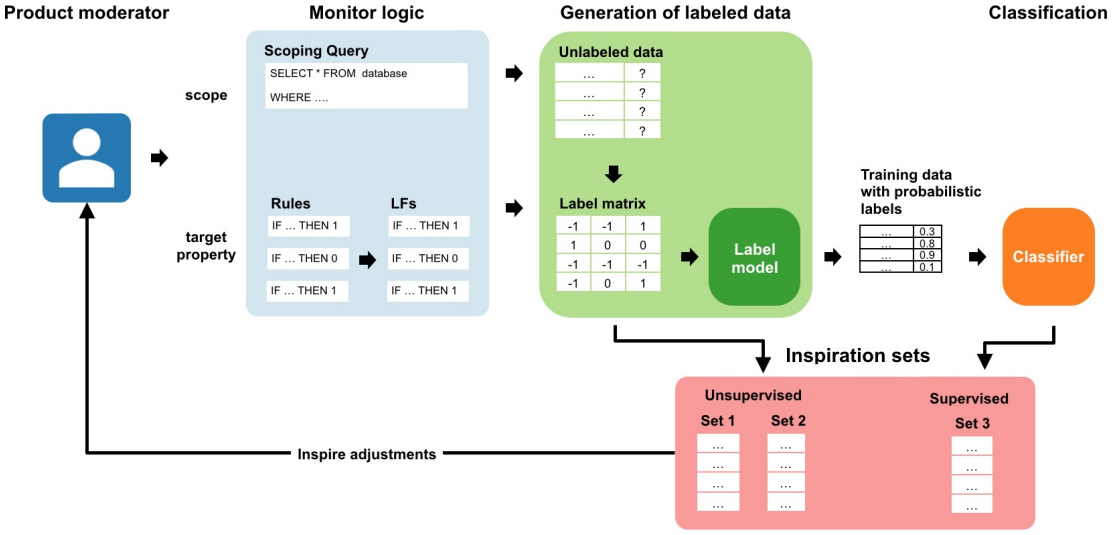


Figure 5.1: This figure shows at which steps in a monitor's pipeline the different types of inspiration sets were drawn.

Inspiration set type \ Monitor	*bont*	*exotisch materiaal*	*magneetballetjes*	*messen*	*rookwaren*	*wegwerpplastic*
Previously unlabeled data	100	100	100	100	100	100
Data on which LFs previously disagreed	49	100	69	4	100	54
Data most mistaken by the classifier	100	59	55	67	60	100

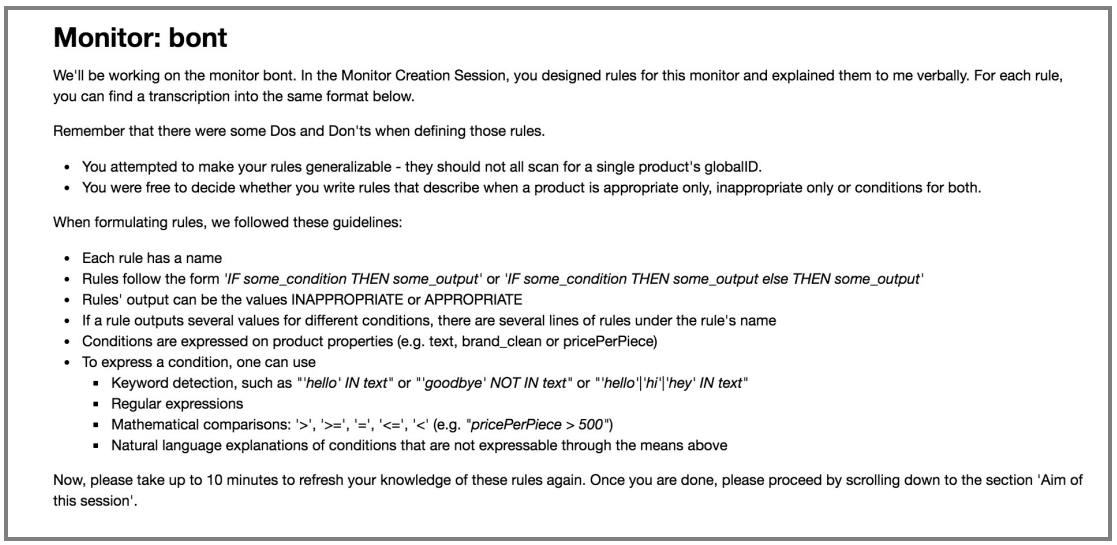
Table 5.1: The amount of data points in each type of inspiration set for every monitor

- **Data most mistaken by the classifier** We trained the whole pipeline using the training set with probabilistic labels from the initial set of LFs. We let the classifier predict the probability of being inappropriate for each data point in the development set. Thereafter, we calculated the difference between this predicted probability and the data point's true label. We divided the data points in the development set into two groups based on their true labels. For each group, we tried to draw the 50 data points for which this difference was the highest. If there were less than 50 data points available for a group, we drew only those that were there.

5.1.2 Monitor improvement procedure

We improved the monitors that the product moderators had created for our first experiment (see Section 4.2.1 for the categories). To improve one of these monitors, we had a 30-minute session with the product moderator who had created the initial set of LFs

for that monitor.
The setup during a monitor improvement session was as follows: We presented a Jupyter Notebook[1] to the product moderator. In this notebook, the first cells contained a recap of the if-condition-then-outcome structure that had been used to formulate the product moderator's domain knowledge as a rule (see Figure 5.2).



Monitor: bont

We'll be working on the monitor bont. In the Monitor Creation Session, you designed rules for this monitor and explained them to me verbally. For each rule, you can find a transcription into the same format below.

Remember that there were some Dos and Don'ts when defining those rules.

- You attempted to make your rules generalizable - they should not all scan for a single product's globalID.
- You were free to decide whether you write rules that describe when a product is appropriate only, inappropriate only or conditions for both.

When formulating rules, we followed these guidelines:

- Each rule has a name
- Rules follow the form *'IF some_condition THEN some_output'* or *'IF some_condition THEN some_output else THEN some_output'*
- Rules' output can be the values INAPPROPRIATE or APPROPRIATE
- If a rule outputs several values for different conditions, there are several lines of rules under the rule's name
- Conditions are expressed on product properties (e.g. text, brand_clean or pricePerPiece)
- To express a condition, one can use
  - Keyword detection, such as "*'hello' IN text*" or "*'goodbye' NOT IN text*" or "*'hello'|'hi'|'hey' IN text*"
  - Regular expressions
  - Mathematical comparisons: '>', '>=', '=', '<=', '<' (e.g. "*pricePerPiece > 500*")
  - Natural language explanations of conditions that are not expressable through the means above

Now, please take up to 10 minutes to refresh your knowledge of these rules again. Once you are done, please proceed by scrolling down to the section 'Aim of this session'.

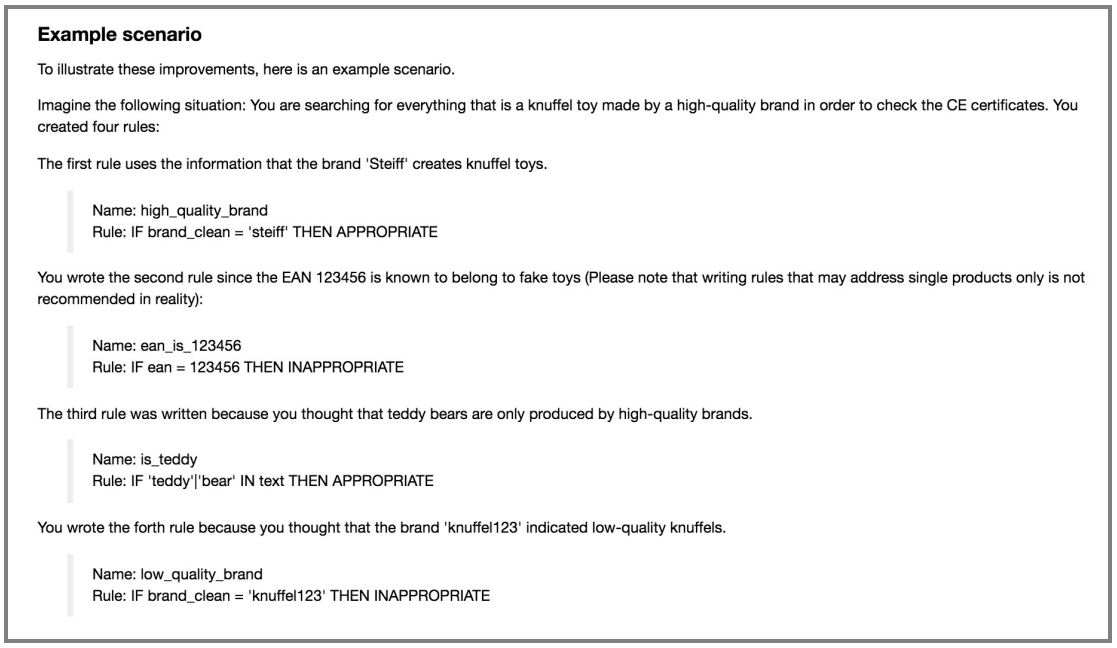
Figure 5.2: A screenshot showing the explanation of the transcriptions of heuristics into a rule format, taken from an inspiration notebook for the monitor *bont*.

After the explanation of rule formulation, we gave a short description of the product attributes that the rules had been based on (e.g. the text or the price per piece). Then, we showed the rules that the product moderator had written in the monitor creation session. We asked the product moderator to inspect these rules for up to 10 minutes to refresh their knowledge of them.
Next, we described the aim of the session, telling the product moderator that they were to improve the rules they had previously defined. We explicitly stated that improving could denote creating a new rule or adjusting or deleting an existing one. To illustrate such changes, we showed them a short fictional example scenario (see Figure 5.3).

We provided the product moderator with examples for possible changes. Figure 5.4 shows the example for creating a new rule. Figure 5.5 shows examples for adjusting or deleting existing rules. After these explanations, the notebook displayed an inspiration set and briefly explained the strategy according to which the set had been drawn. We asked the product moderator to improve the present monitor's rules based on this set. We explicitly stated that they were not allowed to use information or domain knowledge that was not apparent in this set. They were allowed to inspect the offering page of a product in the inspiration set on bol.com and on an internal platform they used for product moderation.

We repeated this improvement procedure three times for every monitor, once for every inspiration set. Originally, we had aimed to randomize the order in which inspiration sets were shown in the improvement sessions. Due to time restrictions, the product moderators had to label the development data later than planned. As labeled data was required to draw the third, supervised inspiration set type (classifier-based strategy),

[1] https://jupyter.org/



Example scenario

To illustrate these improvements, here is an example scenario.

Imagine the following situation: You are searching for everything that is a knuffel toy made by a high-quality brand in order to check the CE certificates. You created four rules:

The first rule uses the information that the brand 'Steiff' creates knuffel toys.

> Name: high_quality_brand
> Rule: IF brand_clean = 'steiff' THEN APPROPRIATE

You wrote the second rule since the EAN 123456 is known to belong to fake toys (Please note that writing rules that may address single products only is not recommended in reality):

> Name: ean_is_123456
> Rule: IF ean = 123456 THEN INAPPROPRIATE

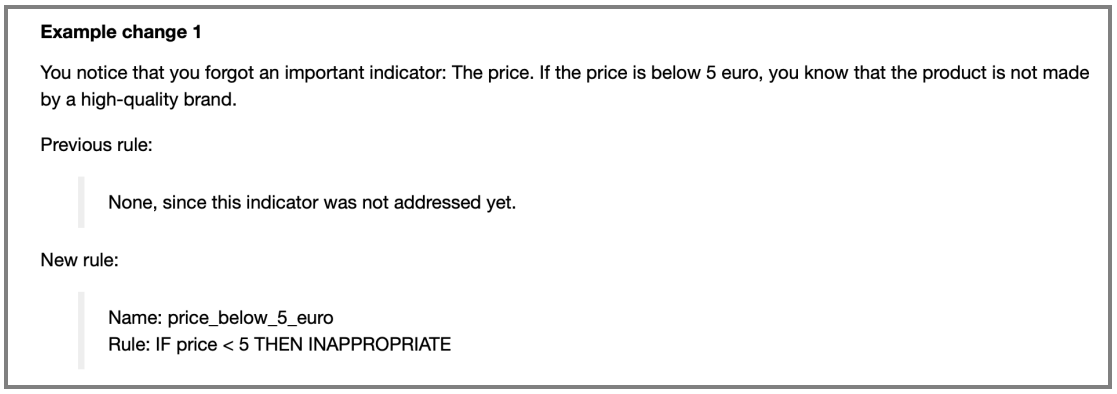
The third rule was written because you thought that teddy bears are only produced by high-quality brands.

> Name: is_teddy
> Rule: IF 'teddy'|'bear' IN text THEN APPROPRIATE

You wrote the forth rule because you thought that the brand 'knuffel123' indicated low-quality knuffels.

> Name: low_quality_brand
> Rule: IF brand_clean = 'knuffel123' THEN INAPPROPRIATE

Figure 5.3: A screenshot showing the fictional example scenario, taken from an inspiration notebook for the monitor *bont*.



Example change 1

You notice that you forgot an important indicator: The price. If the price is below 5 euro, you know that the product is not made by a high-quality brand.

Previous rule:

> None, since this indicator was not addressed yet.

New rule:

> Name: price_below_5_euro
> Rule: IF price < 5 THEN INAPPROPRIATE

Figure 5.4: A screenshot showing the creation of a new rule in the fictional example scenario. The screenshot was taken from an inspiration notebook for the monitor *bont*.

we had to postpone the improvement sessions using this set. In consequence, only the order of the first (abstain-based strategy) and the second (disagreement-based strategy) inspiration set type were randomized across monitors, but the third inspiration set was always used in the last monitor improvement session.

5.1.3 Evaluation procedure

For answering research questions RQ3a, RQ3b and RQ3c, we used analyses explained in Chapter 4. For each inspiration set type, we investigated the change in the number of labels per sample, in the probabilities of the label *inappropriate*, in the LFs' statistics and in LFs' effect on the label model.
For answering research question RQ4, we used the monitors' test sets. We contrasted

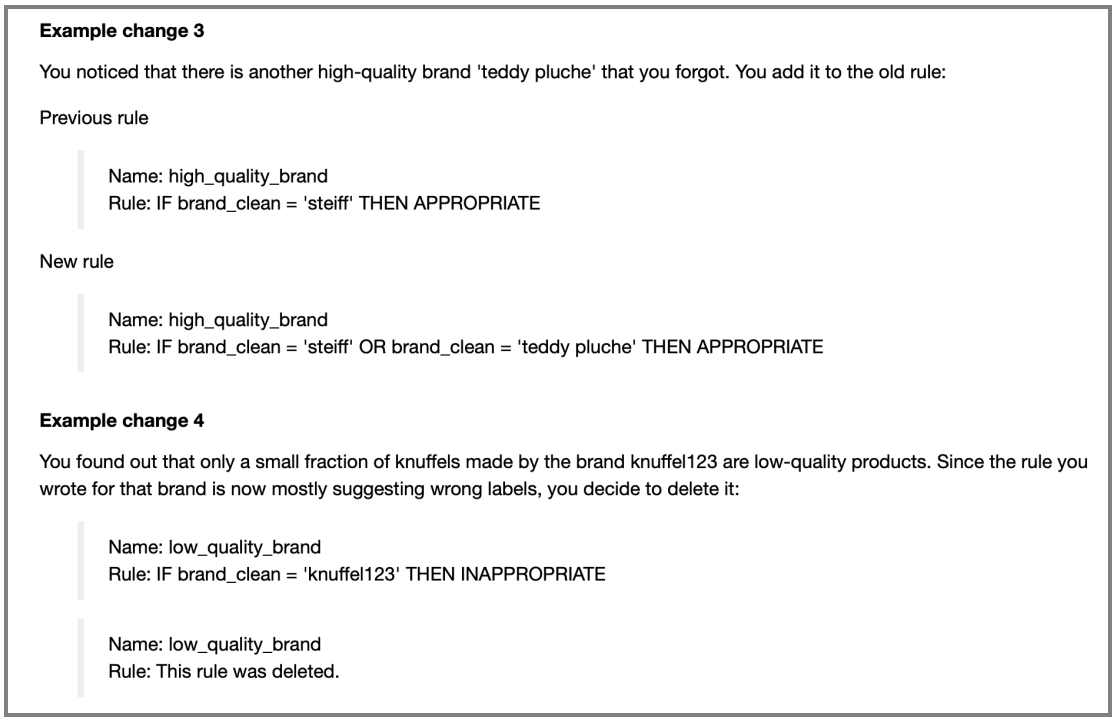


Figure 5.5: A screenshot showing the adjustment of an existing rule and the deletion of an existing rule in the fictional example scenario. The screenshot was taken from an inspiration notebook for the monitor *bont*.

the performance of the label model and that of the whole pipeline (i.e. the classifier) for the initial and all three adjusted versions of a monitor.

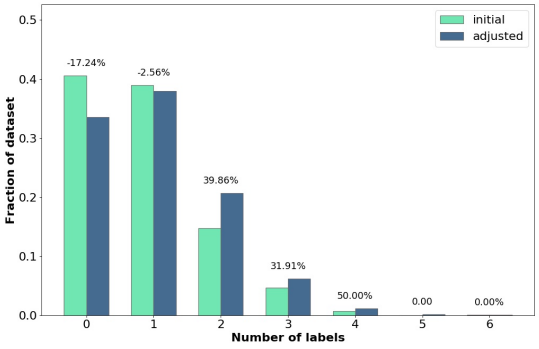
5.2 Results

In the following subsections, we subsequently report the results for each inspiration set type. Last, we report the results on the test data.

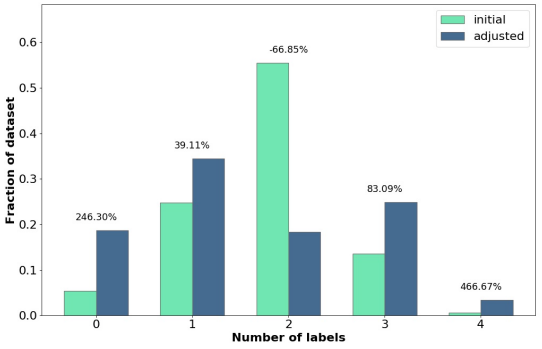
5.2.1 Adjustments inspired by previously unlabeled data

5.2.1.1 Number of labels per sample

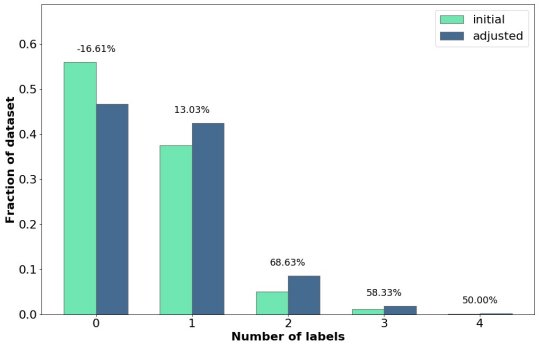
For every monitor, Figure 5.6 shows which fraction of the training set received how many labels per sample.
Across the monitors, almost all fractions' sizes changed in comparison to the results obtained with the initial LF sets. The fractions with more labels per sample tended to increase.
In all monitors except for *exotisch materiaal*, the amount of unlabeled data considerably decreased by 16.61 (*magneetballetjes*) to 48.90 (*wegwerpplastic*)%. In these monitors, most of the samples that got labeled received a label from only one LF. Again, there was a downward trend in the dataset fractions: As the number of labels per sample increased, the dataset fraction receiving this number of labels decreased. The difference



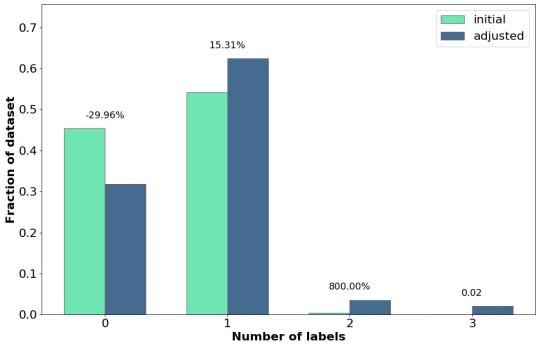
(a) *Bont*



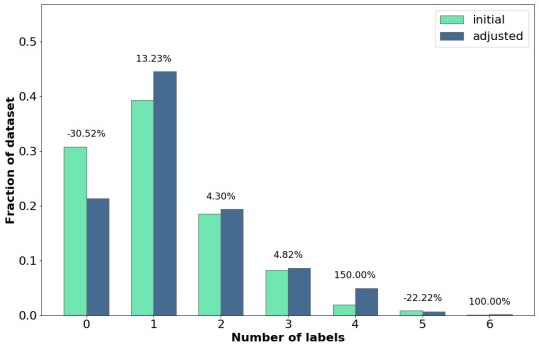
(b) *Exotisch materiaal*



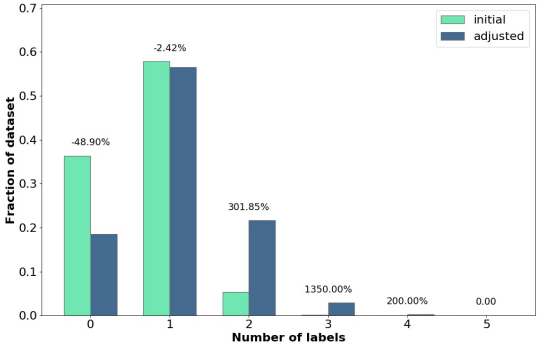
(c) *Magneetballetjes*



(d) *Messen*



(e) *Rookwaren*

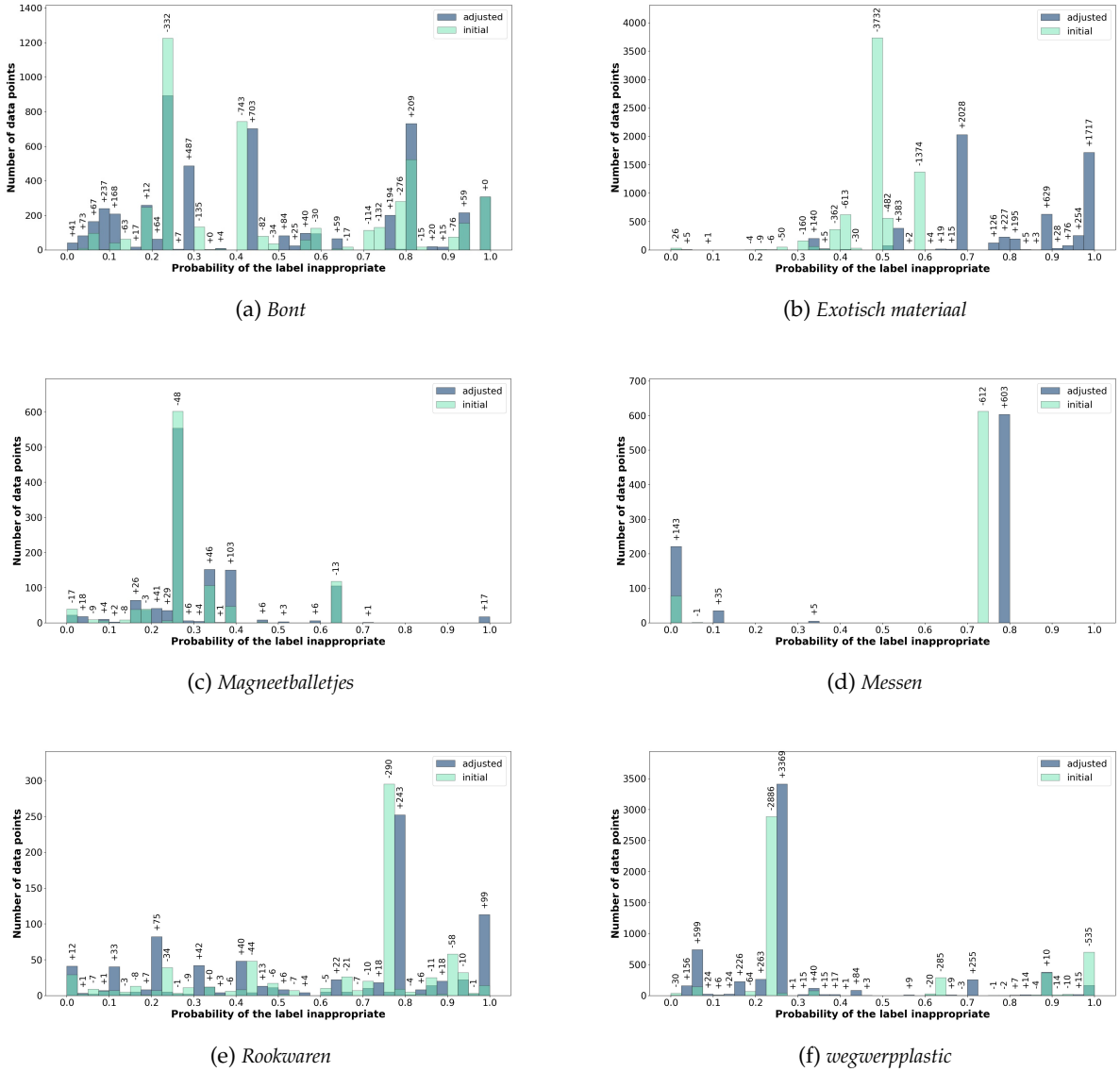


(f) *wegwerpplastic*

Figure 5.6: The change in fractions of the training sets that received a certain number of labels per sample between using the initial and the adjusted LFs.

in fractions again diminished the higher the number of labels was. Furthermore, the maximum number of labels per sample increased by one in *messen*.
The monitor *exotisch materiaal* stood out, because the amount of unlabeled data increased by 246.30%. In this monitor, most labeled training samples received two or more labels.

5.2.1.2 Probability of the label inappropriate



(a) *Bont*

(b) *Exotisch materiaal*

(c) *Magneetballetjes*

(d) *Messen*

(e) *Rookwaren*

(f) *wegwerpplastic*

Figure 5.7: The change in the distribution of probabilities for the label *inappropriate* over the training samples as suggested by the label model trained on the whole set of LFs for each monitor. The change is shown between using the initial and the adjusted set of LFs. The probabilities are divided into 40 bins of size 0.025 each.

For every monitor, Figure 5.7 shows the distribution of the probabilities of the label *inappropriate* across the training samples. All the assigned probabilities for the label *inappropriate* cover most of the available range between 0 and 1 in all monitors besides

messen (where the highest probability had been close to 0.8). In the monitor *messen*, the label model output only four different probability values across the training samples. Besides variations in the sizes of nearby bins, we see general probability changes after the adjustments. In *exotisch materiaal*, the assigned probabilities shifted towards higher values after the adjustments. In *messen*, the assigned probabilities shifted more towards both ends of the range of possible values. In *wegwerpplastic*, there was an increase in low probability values and a decrease in high ones. In *bont*, *magneetballetjes* and *rookwaren* the general tendencies of the values stayed the same.

5.2.1.3 LFs' statistics and effects on the label model

Performance \ Monitor	*bont*	*exotisch materiaal*	*magneetballetjes*	*messen*	*rookwaren*	*wegwerpplastic*
Recall	0.96	1.00	0.71	0.06	0.75	0.65
Precision	0.41	0.04	0.23	0.01	0.10	0.83
F2	0.76	0.18	0.50	0.03	0.33	0.68
Recall (initial)	0.98	0.00	0.57	0.11	0.67	0.60
Precision (initial)	0.42	0.00	0.17	0.02	0.10	0.54
F2 (initial)	0.78	0.00	0.39	0.06	0.31	0.59

Table 5.2: The performance of the label model on the validation set using the complete set of LFs across all monitors.

Table 5.2 shows the performance of the label model on the validation set when all LFs were used before (initial) and after the adjustments. We present this table to allow for comparisons with individual LFs' effects on the label model's performance, which were also calculated on the validation set. For each LF, Table 5.3 shows the statistics, how it had been changed, as well as both the effects on the label model before and after the adjustments. For newly added LFs, there were no effects before the adjustments.
In comparison to their initial performance, the label models in *exotisch materiaal*, *magneetballetjes* and *rookwaren* showed a clear recall increase and those of the former two monitors also slightly improved in precision. The label model in *wegwerpplastic* strongly improved in precision while also increasing the recall. In the other two monitors, all performance scores slightly decreased.
Overall, the effects of LFs that had initially had zero coverage remained mostly unchanged. Also, in most LFs whose effect on the performance had initially been less than 1.00%, there was no perceivable change of this effect.
In all monitors except for *messen* and *rookwaren*, the domain experts adjusted existing LFs. After most of these adjustments, the LFs' change of effects on the label models' performance had been small. Some of the changes had led to positive, some to negative changes in these effects. A strong positive change in the effect on the performance could

be perceived in LF 0 from *magneetballetjes*.
As for newly added LFs, most of them had a small effect on the performance, and most of these effects were positive. In *rookwaren*, one newly added LF (LF 17) had a clear positive effect on the performance. One newly added LF from *messen* (LF 8) had a strong negative effect on the performance.
In most of the existing LFs that had not been adjusted, the effects on the label models' performance slightly changed (in positive or negative direction). Two changes in effects of these LFs were worth noting. First, the exclusion of LFs whose removal had previously led to a 100.00% performance decrease (LF 1 from *messen* and LF 3 from *magneetballetjes*) now led to a strong performance rise. Second, the strong negative effect on the performance of LF 3 from *rookwaren* clearly decreased.

Index	Change	Polarity	Coverage	Overlaps	Conflicts	Correct	Incorrect	%Effect on LM			% Effect on LM (initial)		
								Recall	Precision	F2	Recall	Precision	F2
0	/	[0]	0.25	0.14	0.06	77	1	1.89	3.38	2.45	0.00	3.03	1.14
1	/	[0]	0.01	0.01	0.01	3	0	0.00	-0.78	-0.29	0.00	-1.59	-0.58
2	/	/	0.00	0.00	0.00	0	0	0.00	0.00	0.00	0.00	0.00	0.00
3	/	[0]	0.02	0.02	0.00	6	0	0.00	0.00	0.00	0.00	0.00	0.00
4	/	[0]	0.03	0.03	0.01	9	3	0.00	0.00	0.00	0.00	0.00	0.00
5	/	[0]	0.01	0.00	0.00	3	0	0.00	0.00	0.00	0.00	0.00	0.00
6	/	[0]	0.21	0.12	0.07	79	5	0.00	0.00	0.00	0.00	3.03	1.14
7	/	[0, 1]	0.02	0.02	0.01	2	5	3.77	-3.44	1.23	3.70	-1.03	2.01
8	/	[1]	0.00	0.00	0.00	0	0	0.00	0.00	0.00	0.00	0.00	0.00
9	/	[1]	0.13	0.10	0.07	11	34	7.55	-31.06	-3.75	5.56	-27.25	-4.34
10	/	[1]	0.02	0.02	0.01	5	1	0.00	-3.20	-1.16	0.00	-0.79	-0.29
11	/	[1]	0.00	0.00	0.00	4	0	0.00	0.00	0.00	0.00	0.00	0.00
12	/	[1]	0.00	0.00	0.00	4	0	0.00	0.00	0.00	0.00	0.00	0.00
13	A	[0, 1]	0.16	0.10	0.07	46	48	69.81	22.11	60.98	70.37	25.64	61.95
14	/	/	0.00	0.00	0.00	0	0	0.00	0.00	0.00	0.00	0.00	0.00
15	N	[0]	0.01	0.01	0.00	4	0	0.00	0.77	0.29	/	/	/
16	N	[0]	0.14	0.08	0.03	40	2	-1.89	-4.31	-2.77	/	/	/
17	N	[1]	0.03	0.02	0.02	0	5	0.00	0.00	0.00	/	/	/
18	N	[0]	0.00	0.00	0.00	1	0	0.00	0.00	0.00	/	/	/

(a) Bont

Index	Change	Polarity	Coverage	Overlaps	Conflicts	Correct	Incorrect	%Effect on LM			Effect on LM (initial)		
								Recall	Precision	F2	Recall	Precision	F2
0	A	[0, 1]	0.73	0.46	0.39	10	206	50.00	-237.14	-84.00	-0.50	-0.02	-0.09
1	/	[1]	0.00	0.00	0.00	2	0	0.00	0.00	0.00	0.00	0.00	0.00
2	/	[1]	0.10	0.09	0.05	7	32	0.00	-4.89	-4.15	-1.00	-0.04	-0.19
3	/	[0]	0.02	0.02	0.02	10	0	0.00	0.84	0.72	-1.00	-0.05	-0.21
4	A	[0]	0.00	0.00	0.00	0	0	0.00	0.00	0.00	-1.00	-0.05	-0.20
5	A	[0]	0.34	0.32	0.32	57	3	0.00	-3.96	-3.37	-1.00	-0.04	-0.19
6	A	[0]	0.12	0.10	0.10	43	2	0.00	0.84	0.72	-1.00	-0.05	-0.19
7	N	[1]	0.26	0.26	0.21	6	5	0.00	1.26	1.08	/	/	/

(b) Exotisch materiaal

Index	Change	Polarity	Coverage	Overlaps	Conflicts	Correct	Incorrect	%Effect on LM			% Effect on LM (initial)		
								Recall	Precision	F2	Recall	Precision	F2
0	A	[1]	0.01	0.00	0.00	4	0	20.00	16.19	18.37	0.00	0.00	0.00
1	A	[0]	0.04	0.02	0.00	10	0	0.00	4.35	1.96	0.00	0.00	0.00
2	A	[0]	0.32	0.08	0.03	114	0	0.00	31.25	16.67	0.00	32.35	17.74
3	/	[1]	0.08	0.04	0.03	4	30	20.00	-252.00	-21.21	100.00	100.00	100.00
4	A	[0]	0.08	0.03	0.01	25	0	0.00	0.00	0.00	0.00	0.00	0.00
5	A	[0]	0.03	0.02	0.01	6	0	0.00	0.00	0.00	0.00	0.00	0.00
6	N	[0]	0.11	0.04	0.01	24	0	0.00	4.35	1.96	/	/	/
7	N	[1]	0.01	0.00	0.00	1	1	0.00	0.00	0.00	/	/	/

(c) Magneetballetjes

Index	Change	Polarity	Coverage	Overlaps	Conflicts	Correct	Incorrect	%Effect on LM			% Effect on LM (initial)		
								Recall	Precision	F2	Recall	Precision	F2
0	/	/	0.00	0.00	0.00	0	0	0.00	0.00	0.00	0.00	0.00	0.00
1	/	[1]	0.49	0.01	0.01	2	96	-100.00	-5966.67	-334.67	100.00	100.00	100.00
2	/	[1]	0.00	0.00	0.00	0	0	0.00	0.00	0.00	0.00	0.00	0.00
3	/	/	0.00	0.00	0.00	0	0	0.00	0.00	0.00	0.00	0.00	0.00
4	/	[0]	0.04	0.01	0.00	11	0	0.00	2.15	1.21	0.00	2.04	1.18
5	/	[0]	0.02	0.02	0.00	4	0	0.00	0.00	0.00	0.00	0.00	0.00
6	N	/	0.00	0.00	0.00	0	0	0.00	0.00	0.00	/	/	/
7	N	[0]	0.06	0.03	0.00	10	0	0.00	0.00	0.00	/	/	/
8	N	[0]	0.03	0.01	0.01	4	1	-100.00	-91.58	-95.21	/	/	/
9	N	[1]	0.01	0.00	0.00	2	1	0.00	0.00	0.00	/	/	/
10	N	[0]	0.11	0.04	0.00	0	0	0.00	0.00	0.00	/	/	/

(d) Messen

Index	Change	Polarity	Coverage	Overlaps	Conflicts	Correct	Incorrect	%Effect on LM			% Effect on LM (initial)		
								Recall	Precision	F2	Recall	Precision	F2
0	/	[1]	0.42	0.22	0.13	9	65	11.11	-59.64	-24.63	0.00	-51.85	-27.45
1	/	[1]	0.01	0.01	0.01	0	0	0.00	0.00	0.00	0.00	0.00	0.00
2	/	[1]	0.11	0.07	0.04	6	13	0.00	0.00	0.00	0.00	0.00	0.00
3	/	[0]	0.05	0.05	0.03	5	2	-11.11	-8.64	-9.50	-25.00	-12.64	-16.91
4	/	[0]	0.09	0.06	0.05	15	0	0.00	0.00	0.00	0.00	6.82	4.41
5	/	[0]	0.04	0.04	0.02	8	1	0.00	2.22	1.45	0.00	2.38	1.52
6	/	[0]	0.02	0.01	0.00	6	0	0.00	1.12	0.73	0.00	1.20	0.76
7	/	[0]	0.04	0.03	0.03	3	0	0.00	-1.15	-0.74	0.00	3.53	2.26
8	/	[0]	0.03	0.03	0.02	2	1	0.00	0.00	0.00	0.00	0.00	0.00
9	/	[0]	0.00	0.00	0.00	3	0	0.00	0.00	0.00	0.00	0.00	0.00
10	/	[0]	0.08	0.08	0.06	16	0	0.00	-1.15	-0.74	0.00	4.65	2.99
11	/	[0]	0.00	0.00	0.00	1	0	0.00	0.00	0.00	0.00	0.00	0.00
12	/	[0]	0.01	0.01	0.01	1	0	0.00	0.00	0.00	0.00	0.00	0.00
13	/	[1]	0.01	0.01	0.01	2	3	-11.11	-9.86	-10.30	-12.50	-11.14	-11.64
14	/	[1]	0.01	0.01	0.00	1	1	0.00	0.00	0.00	0.00	0.00	0.00
15	/	[1]	0.17	0.14	0.09	4	33	22.22	4.94	11.85	12.50	-1.06	4.41
16	N	[0]	0.02	0.00	0.00	0	0	0.00	0.00	0.00	/	/	/
17	N	[0, 1]	0.20	0.14	0.07	23	16	11.11	10.09	10.45	/	/	/
18	N	[0]	0.01	0.00	0.00	2	0	0.00	0.00	0.00	/	/	/

(e) Rookwaren

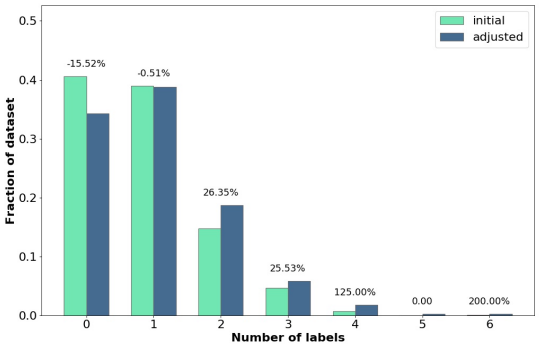
Index	Change	Polarity	Coverage	Overlaps	Conflicts	Correct	Incorrect	%Effect on LM			% Effect on LM (initial)		
								Recall	Precision	F2	Recall	Precision	F2
0	/	[1]	0.02	0.01	0.01	17	1	5.19	-0.19	4.35	8.45	3.32	7.38
1	/	[0]	0.06	0.04	0.01	30	2	0.00	0.00	0.00	2.82	-25.77	-2.26
2	/	/	0.00	0.00	0.00	0	0	0.00	0.00	0.00	0.00	0.00	0.00
3	/	/	0.00	0.00	0.00	0	0	0.00	0.00	0.00	0.00	0.00	0.00
4	/	/	0.00	0.00	0.00	0	0	0.00	0.00	0.00	0.00	0.00	0.00
5	A	[0]	0.69	0.23	0.05	268	9	0.00	3.12	0.53	8.45	-54.93	-0.54
6	A	[0]	0.14	0.13	0.00	60	0	-1.30	3.87	-0.41	0.00	2.94	0.66
7	A	[0]	0.01	0.01	0.00	5	0	0.00	0.00	0.00	0.00	-1.54	-0.33
8	/	[0]	0.01	0.01	0.00	0	0	0.00	0.00	0.00	0.00	0.00	0.00
9	/	[0]	0.01	0.00	0.00	1	1	0.00	0.00	0.00	1.41	0.66	1.24
10	/	[1]	0.01	0.01	0.01	2	4	0.00	-2.20	-0.36	0.00	-2.33	-0.50
11	/	[0]	0.06	0.04	0.00	21	4	0.00	0.00	0.00	4.23	-11.88	1.11
12	/	[0]	0.00	0.00	0.00	0	0	0.00	0.00	0.00	0.00	0.00	0.00
13	A	[1]	0.11	0.06	0.05	73	17	76.62	-3.53	73.21	66.20	42.05	62.81
14	N	[0]	0.00	0.00	0.00	0	0	0.00	0.00	0.00	/	/	/

(f) Wegwerpplastic

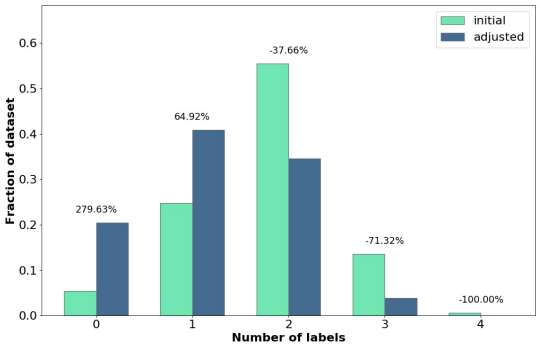
Table 5.3: For each LF in each monitor, this table shows the change, i.e. whether it was adjusted (A), newly added (N) or unchanged (/). Further, the table shows the LFs' statistics on the training set (polarity, coverage overlaps, conflicts), the number of incorrect and correct suggestions on the validation set and their effect on the label model's (LM) performance before (initial) and after the adjustments. Gray lines indicate LFs with zero coverage.

5.2.2 Adjustments inspired by data on which LFs previously disagreed

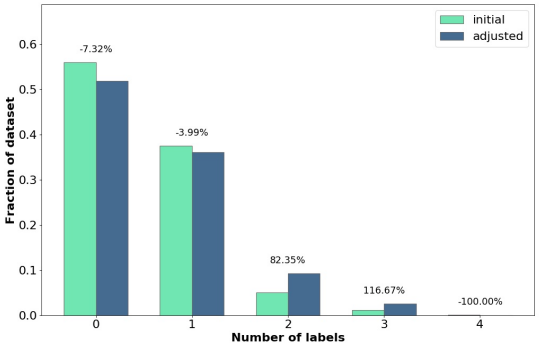
5.2.2.1 Number of labels per sample



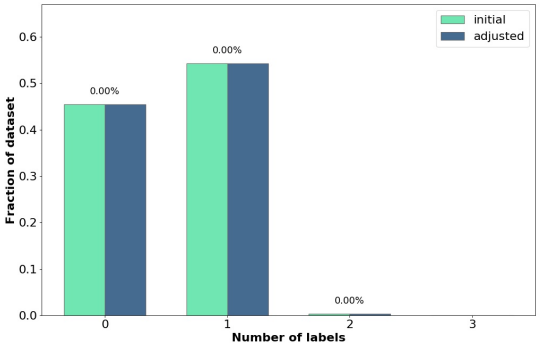
(a) *Bont*



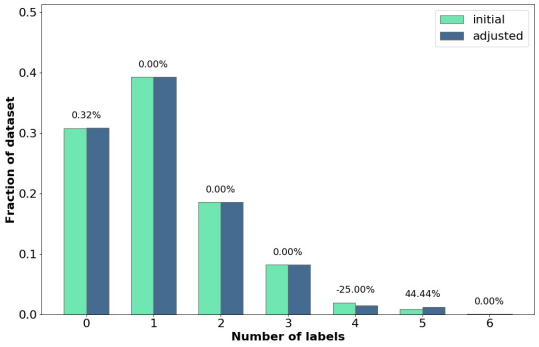
(b) *Exotisch materiaal*



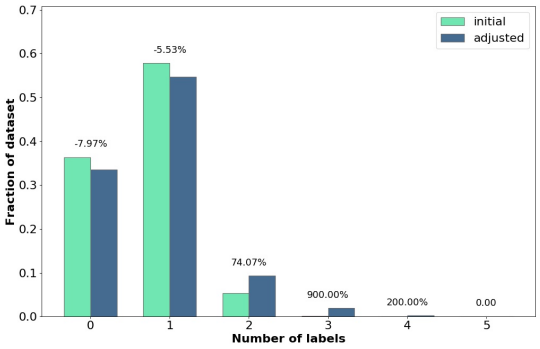
(c) *Magneetballetjes*



(d) *Messen*



(e) *Rookwaren*



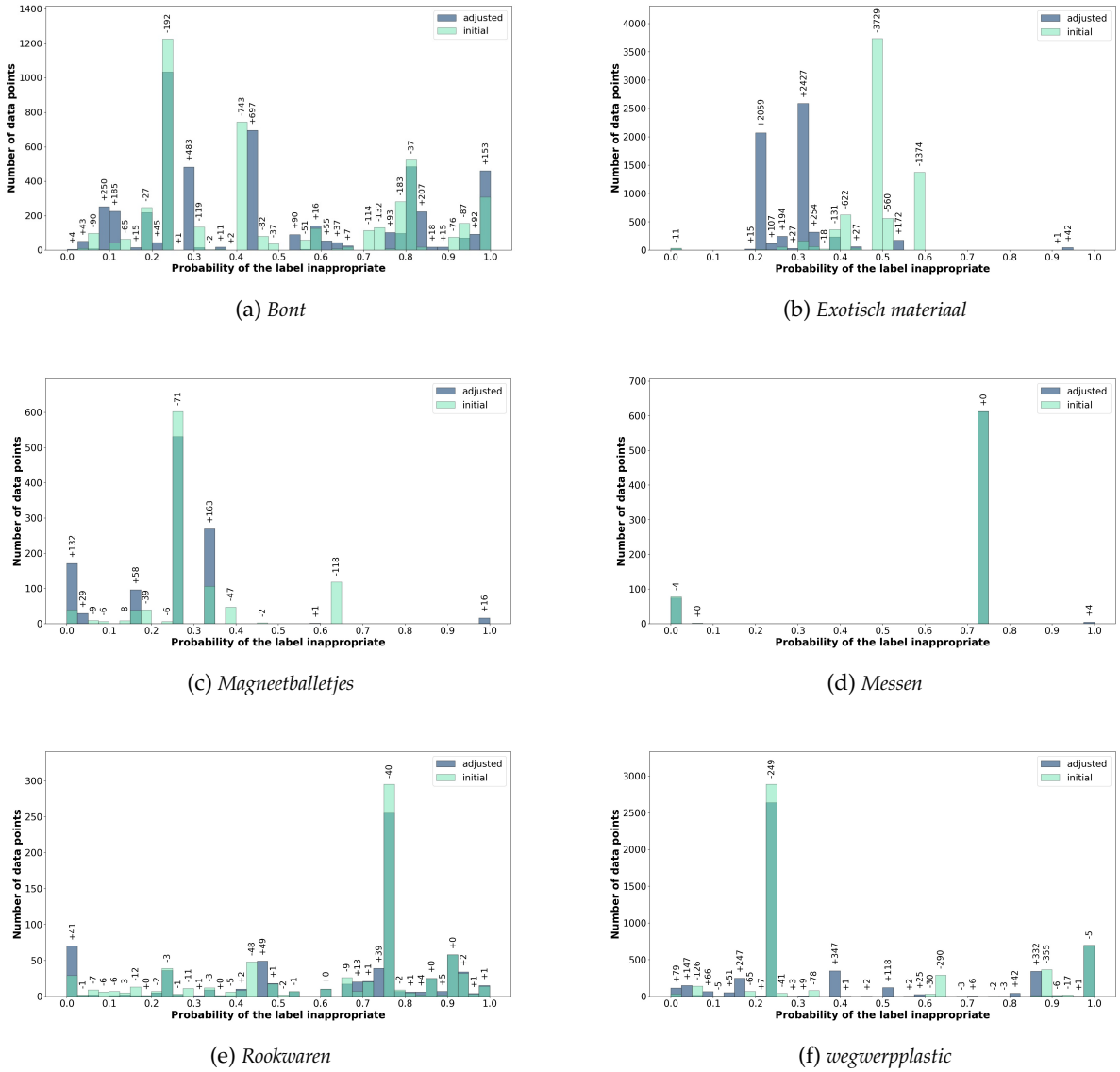
(f) *wegwerpplastic*

Figure 5.8: The change in fractions of the training sets that received a certain number of labels per sample between using the initial and the adjusted LFs.

For every monitor, Figure 5.8 shows which fraction of the training set received how many labels per sample.
Across four monitors, almost all fractions' sizes changed in comparison to the initial results and the fractions with more labels per sample tended to increase. By contrast,

most fractions' sizes in the monitors *messen* and *rookwaren* stayed the same.
In *bont*, *magneetballetjes* and *wegwerpplastic*, the amount of unlabeled data decreased and the number of samples with higher numbers of labels increased.
In *exotisch materiaal*, the amount of data with more than one label decreased, while the amount that received one label at most increased.
All monitors showed the downward trend in the dataset fractions as the number of labels per sample increased. In all monitors, most samples that received a label received only one.

5.2.2.2 Probability of the label inappropriate



(a) *Bont*

(b) *Exotisch materiaal*

(c) *Magneetballetjes*

(d) *Messen*

(e) *Rookwaren*

(f) *wegwerpplastic*

Figure 5.9: The change in the distribution of probabilities for the label *inappropriate* over the training samples as suggested by the label model trained on the whole set of LFs for each monitor. The change is shown between using the initial and the adjusted set of LFs. The probabilities are divided into 40 bins of size 0.025 each.

For every monitor, Figure 5.9 shows the distribution of the probabilities of the label *inappropriate* across the training samples in a histogram.

The assigned probabilities for the label *inappropriate* range from values close to 0 to values close to 1 in all monitors. Notably, in the monitor *messen*, the label model output only four different probability values across the training samples.
In the monitor *exotisch materiaal*, the assigned probabilities shifted towards smaller values than initially. The overall tendencies of the other monitors' label models stayed the same. They were even clearer due to bins' shifts towards the ends of the possible range of values. Importantly, in contrast to initial results, the label models of the monitors *exotisch materiaal*, *magneetballetjes* and *messen* assigned very high probabilities of being *inappropriate* after the adjustments (though only to a small number of samples).

5.2.2.3 LFs' statistics and effects on the label model

Performance \ Monitor	*bont*	*exotisch materiaal*	*magneetballetjes*	*messen*	*rookwaren*	*wegwerpplastic*
Recall	0.98	0.00	0.57	0.11	0.83	0.58
Precision	0.40	0.00	1.00	0.02	0.12	0.57
F2	0.76	0.00	0.63	0.06	0.38	0.58
Recall (initial)	0.98	0.00	0.57	0.11	0.67	0.60
Precision (initial)	0.42	0.00	0.17	0.02	0.10	0.54
F2 (initial)	0.78	0.00	0.39	0.06	0.31	0.59

Table 5.4: The performance of the label model on the validation set using the complete set of LFs across all monitors.

Table 5.4 shows the performance of the label model on the validation set when all LFs were used before (initial) and after the adjustments. We present this table to allow for comparisons with individual LFs' effects on the label model's performance, which were also calculated on the validation set. For each LF, Table 5.5 shows the statistics, how it had been changed, as well as both the effects on the label model before and after the adjustments. For newly added LFs, there were no effects before the adjustments. Please note that in the monitor *exotisch materiaal*, the domain expert had removed LF 2.
The performance of the label models in *exotisch materiaal* and *messen* did not change. In *rookwaren*, the label model strongly improved in recall and slightly improved in precision. In *magneetballetjes*, the label model strongly improved in precision. Also, there were small overall performance decreases in *bont* and *wegwerpplastic*.
Overall, the effects of most LFs that had initially had zero coverage remained unchanged (except for LF 0 from *magneetballetjes*). Also, in most LFs whose effect on the performance had initially been less than 1.00%, there was no perceivable change of this effect.

In all monitors, the domain experts had made adjustments (though only adjusting 1 LF in *bont* and *messen*). After most adjustments, negative effects on the performance when removing that LF decreased. This happened for both weak (LF 0 from *bont*, LF 4 from *messen* and LF 11 from *wegwerpplastic*) and strong (LFs 2 and 3 from *magneetballetjes*) effects on the label models. By contrast, instead of a strong negative, the inclusion of LF 3 from *rookwaren* produced a positive effect on the label model's performance after the adjustments. While including LF 0 from *magneetballetjes* had previously had no perceivable effect on the performance, it led to a strong performance increase after the adjustment.
In all monitors except for *magneetballetjes* and *messen*, the domain experts added new LFs. Most of these LFs' effects on the performance was either smaller than 1.00% or small and negative.
With regard to LFs that had not been adjusted, the effects on the label models' performance mostly only slightly changed in all monitors except for *rookwaren*.

Index	Change	Polarity	Coverage	Overlaps	Conflicts	Correct	Incorrect	%Effect on LM			% Effect on LM (initial)		
								Recall	Precision	F2	Recall	Precision	F2
0	A	[0]	0.25	0.12	0.04	77	1	0.00	0.00	0.00	0.00	3.03	1.14
1	/	[0]	0.01	0.01	0.01	3	0	0.00	-0.75	-0.28	0.00	-1.59	-0.58
2	/	/	0.00	0.00	0.00	0	0	0.00	0.00	0.00	0.00	0.00	0.00
3	/	[0]	0.02	0.02	0.00	6	0	0.00	0.00	0.00	0.00	0.00	0.00
4	/	[0]	0.03	0.03	0.01	9	3	0.00	0.00	0.00	0.00	0.00	0.00
5	/	[0]	0.01	0.00	0.00	3	0	0.00	0.00	0.00	0.00	0.00	0.00
6	/	[0]	0.21	0.12	0.07	79	5	0.00	0.00	0.00	0.00	3.03	1.14
7	/	[0, 1]	0.02	0.02	0.01	2	5	3.70	-0.78	2.05	3.70	-1.03	2.01
8	/	[1]	0.00	0.00	0.00	0	0	0.00	0.00	0.00	0.00	0.00	0.00
9	/	[1]	0.13	0.10	0.07	11	34	5.56	-28.79	-5.10	5.56	-27.25	-4.34
10	/	[1]	0.02	0.02	0.01	5	1	0.00	-0.75	-0.28	0.00	-0.79	-0.29
11	/	[1]	0.00	0.00	0.00	4	0	0.00	0.00	0.00	0.00	0.00	0.00
12	/	[1]	0.00	0.00	0.00	4	0	0.00	0.00	0.00	0.00	0.00	0.00
13	/	[0, 1]	0.16	0.10	0.07	45	49	70.37	28.57	61.89	70.37	25.64	61.95
14	/	/	0.00	0.00	0.00	0	0	0.00	0.00	0.00	0.00	0.00	0.00
15	N	[0]	0.14	0.08	0.03	40	2	0.00	-1.50	-0.57	/	/	/
16	N	[1]	0.01	0.01	0.00	2	0	0.00	0.00	0.00	/	/	/
17	N	[1]	0.02	0.02	0.01	2	11	0.00	-2.27	-0.85	/	/	/

(a) Bont

Index	Change	Polarity	Coverage	Overlaps	Conflicts	Correct	Incorrect	Effect on LM			Effect on LM (initial)		
								Recall	Precision	F2	Recall	Precision	F2
0	A	[0, 1]	0.73	0.38	0.38	10	206	-0.20	-0.18	-0.20	-0.50	-0.02	-0.09
1	/	[1]	0.00	0.00	0.00	2	0	0.00	0.00	0.00	0.00	0.00	0.00
3	A	[0]	0.03	0.02	0.02	12	0	-1.00	-0.04	-0.19	-1.00	-0.05	-0.21
4	A	[0]	0.00	0.00	0.00	0	0	-1.00	-0.04	-0.18	-1.00	-0.05	-0.20
5	A	[0]	0.34	0.31	0.31	57	3	-1.00	-0.05	-0.19	-1.00	-0.04	-0.19
6	/	[0]	0.11	0.09	0.08	39	2	0.00	0.00	0.00	-1.00	-0.05	-0.19
7	N	[0]	0.00	0.00	0.00	1	0	-1.00	-0.04	-0.18	/	/	/

(b) Exotisch materiaal

Index	Change	Polarity	Coverage	Overlaps	Conflicts	Correct	Incorrect	%Effect on LM			% Effect on LM (initial)		
								Recall	Precision	F2	Recall	Precision	F2
0	A	[1]	0.01	0.00	0.00	4	0	75.00	0.00	72.41	0.00	0.00	0.00
1	/	[0]	0.04	0.04	0.00	9	0	0.00	0.00	0.00	0.00	0.00	0.00
2	A	[0]	0.32	0.09	0.00	115	0	0.00	0.00	0.00	0.00	32.35	17.74
3	A	[1]	0.01	0.00	0.00	1	0	0.00	0.00	0.00	100.00	100.00	100.00
4	A	[0]	0.18	0.06	0.00	58	0	0.00	0.00	0.00	0.00	0.00	0.00
5	A	[0]	0.09	0.08	0.00	25	0	0.00	0.00	0.00	0.00	0.00	0.00

(c) Magneetballetjes

Index	Change	Polarity	Coverage	Overlaps	Conflicts	Correct	Incorrect	%Effect on LM			% Effect on LM (initial)		
								Recall	Precision	F2	Recall	Precision	F2
0	/	/	0.00	0.00	0.00	0	0	0.00	0.00	0.00	0.00	0.00	0.00
1	/	[1]	0.49	0.00	0.00	2	96	100.00	100.00	100.00	100.00	100.00	100.00
2	/	[1]	0.00	0.00	0.00	0	0	0.00	0.00	0.00	0.00	0.00	0.00
3	/	/	0.00	0.00	0.00	0	0	0.00	0.00	0.00	0.00	0.00	0.00
4	A	[0, 1]	0.04	0.00	0.00	9	2	0.00	0.00	0.00	0.00	2.04	1.18
5	/	[0]	0.02	0.00	0.00	4	0	0.00	0.00	0.00	0.00	0.00	0.00

(d) Messen

Index	Change	Polarity	Coverage	Overlaps	Conflicts	Correct	Incorrect	%Effect on LM			% Effect on LM (initial)		
								Recall	Precision	F2	Recall	Precision	F2
0	/	[1]	0.42	0.19	0.12	9	65	10.00	-37.45	-15.34	0.00	-51.85	-27.45
1	A	[1]	0.05	0.05	0.04	1	7	0.00	0.00	0.00	0.00	0.00	0.00
2	/	[1]	0.11	0.07	0.04	6	13	0.00	0.00	0.00	0.00	0.00	0.00
3	A	[0]	0.04	0.03	0.02	5	0	0.00	7.69	5.04	-25.00	-12.64	-16.91
4	/	[0]	0.09	0.06	0.05	15	0	0.00	6.67	4.35	0.00	6.82	4.41
5	/	[0]	0.04	0.03	0.03	8	1	0.00	1.18	0.75	0.00	2.38	1.52
6	/	[0]	0.02	0.02	0.02	6	0	0.00	4.55	2.94	0.00	1.20	0.76
7	/	[0]	0.04	0.03	0.03	3	0	0.00	-1.20	-0.76	0.00	3.53	2.26
8	/	[0]	0.03	0.02	0.02	2	1	0.00	0.00	0.00	0.00	0.00	0.00
9	/	[0]	0.00	0.00	0.00	3	0	0.00	0.00	0.00	0.00	0.00	0.00
10	/	[0]	0.08	0.07	0.07	16	0	0.00	-3.70	-2.33	0.00	4.65	2.99
11	/	[0]	0.00	0.00	0.00	1	0	0.00	0.00	0.00	0.00	0.00	0.00
12	/	[0]	0.01	0.01	0.01	1	0	0.00	0.00	0.00	0.00	0.00	0.00
13	A	[1]	0.01	0.01	0.00	2	2	-10.00	-8.71	-9.17	-12.50	-11.14	-11.64
14	/	[1]	0.01	0.01	0.00	1	1	0.00	0.00	0.00	0.00	0.00	0.00
15	/	[1]	0.17	0.13	0.08	4	33	0.00	-12.00	-7.32	12.50	-1.06	4.41
16	N	/	0.00	0.00	0.00	0	1	0.00	0.00	0.00	/	/	/
17	N	[0]	0.02	0.02	0.02	0	1	0.00	0.00	0.00	/	/	/

(e) Rookwaren

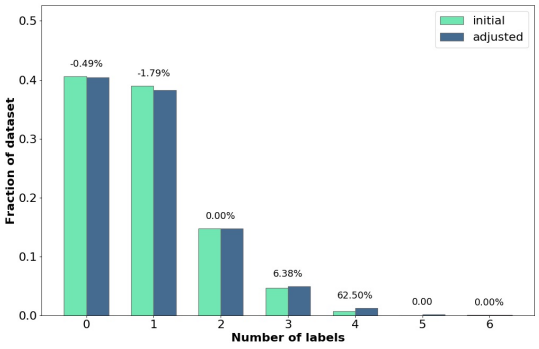
Index	Change	Polarity	Coverage	Overlaps	Conflicts	Correct	Incorrect	%Effect on LM			% Effect on LM (initial)		
								Recall	Precision	F2	Recall	Precision	F2
0	/	[1]	0.02	0.01	0.01	17	1	10.14	3.78	8.92	8.45	3.32	7.38
1	/	[0]	0.06	0.01	0.00	30	2	2.90	-29.11	-2.28	2.82	-25.77	-2.26
2	/	/	0.00	0.00	0.00	0	0	0.00	0.00	0.00	0.00	0.00	0.00
3	/	/	0.00	0.00	0.00	0	0	0.00	0.00	0.00	0.00	0.00	0.00
4	/	/	0.00	0.00	0.00	0	0	0.00	0.00	0.00	0.00	0.00	0.00
5	/	[0]	0.43	0.07	0.01	152	1	5.80	-48.03	-1.75	8.45	-54.93	-0.54
6	/	[0]	0.04	0.03	0.00	20	0	0.00	0.82	0.17	0.00	2.94	0.66
7	/	[0]	0.01	0.00	0.00	5	0	0.00	-0.83	-0.17	0.00	-1.54	-0.33
8	/	[0]	0.01	0.01	0.00	0	0	0.00	0.00	0.00	0.00	0.00	0.00
9	/	[0]	0.01	0.00	0.00	1	1	1.45	0.63	1.28	1.41	0.66	1.24
10	/	[1]	0.01	0.00	0.00	2	4	0.00	-2.54	-0.51	0.00	-2.33	-0.50
11	A	[0]	0.03	0.01	0.00	12	1	1.45	-6.47	-0.07	4.23	-11.88	1.11
12	/	[0]	0.00	0.00	0.00	0	0	0.00	0.00	0.00	0.00	0.00	0.00
13	/	[1]	0.07	0.01	0.01	58	9	68.12	41.55	64.86	66.20	42.05	62.81
14	N	[0]	0.01	0.01	0.00	5	0	0.00	0.00	0.00	/	/	/
15	N	[0]	0.12	0.08	0.02	38	5	0.00	0.82	0.17	/	/	/
16	N	[1]	0.01	0.01	0.01	1	6	0.00	0.00	0.00	/	/	/

(f) Wegwerpplastic

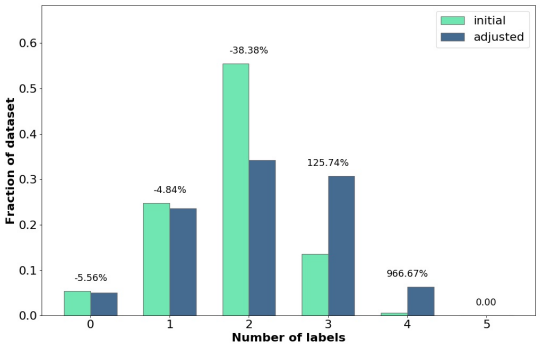
Table 5.5: For each LF in each monitor, this table shows the change, i.e. whether it was adjusted (A), newly added (N) or unchanged (/). Further, the table shows the LFs' statistics on the training set (polarity, coverage overlaps, conflicts), the number of incorrect and correct suggestions on the validation set and their effect on the label model's (LM) performance before (initial) and after the adjustments. Gray lines indicate LFs with zero coverage.

5.2.3 Adjustments inspired by data most mistaken by the classifier

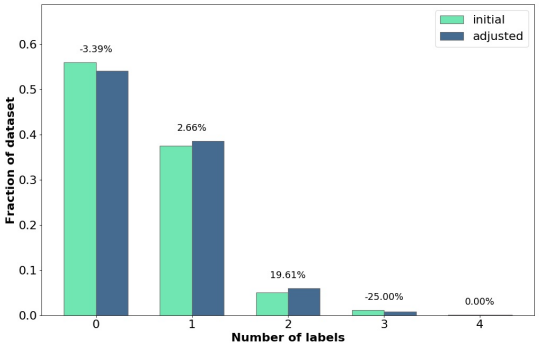
5.2.3.1 Number of labels per sample



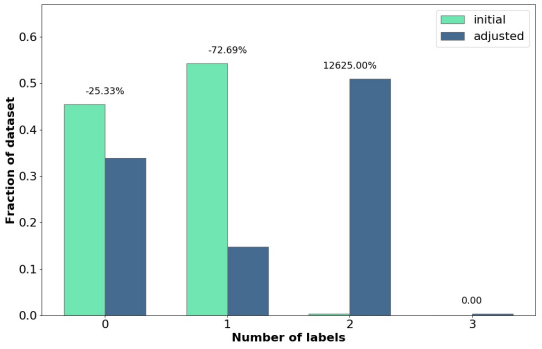
(a) *Bont*



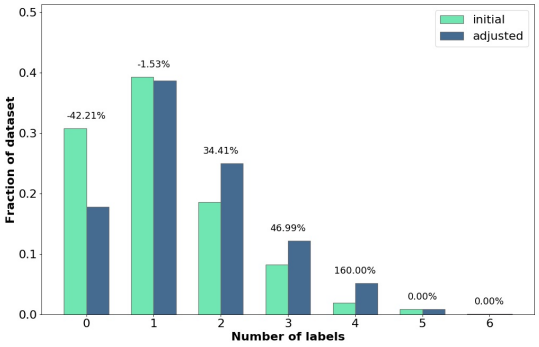
(b) *Exotisch materiaal*



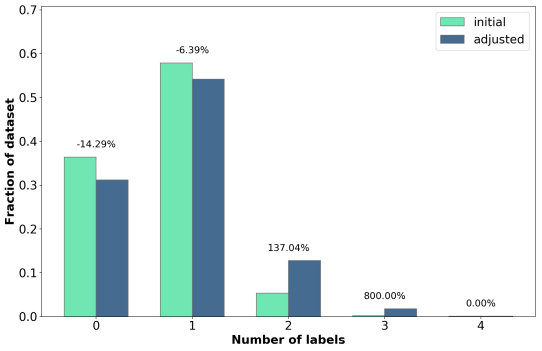
(c) *Magneetballetjes*



(d) *Messen*



(e) *Rookwaren*



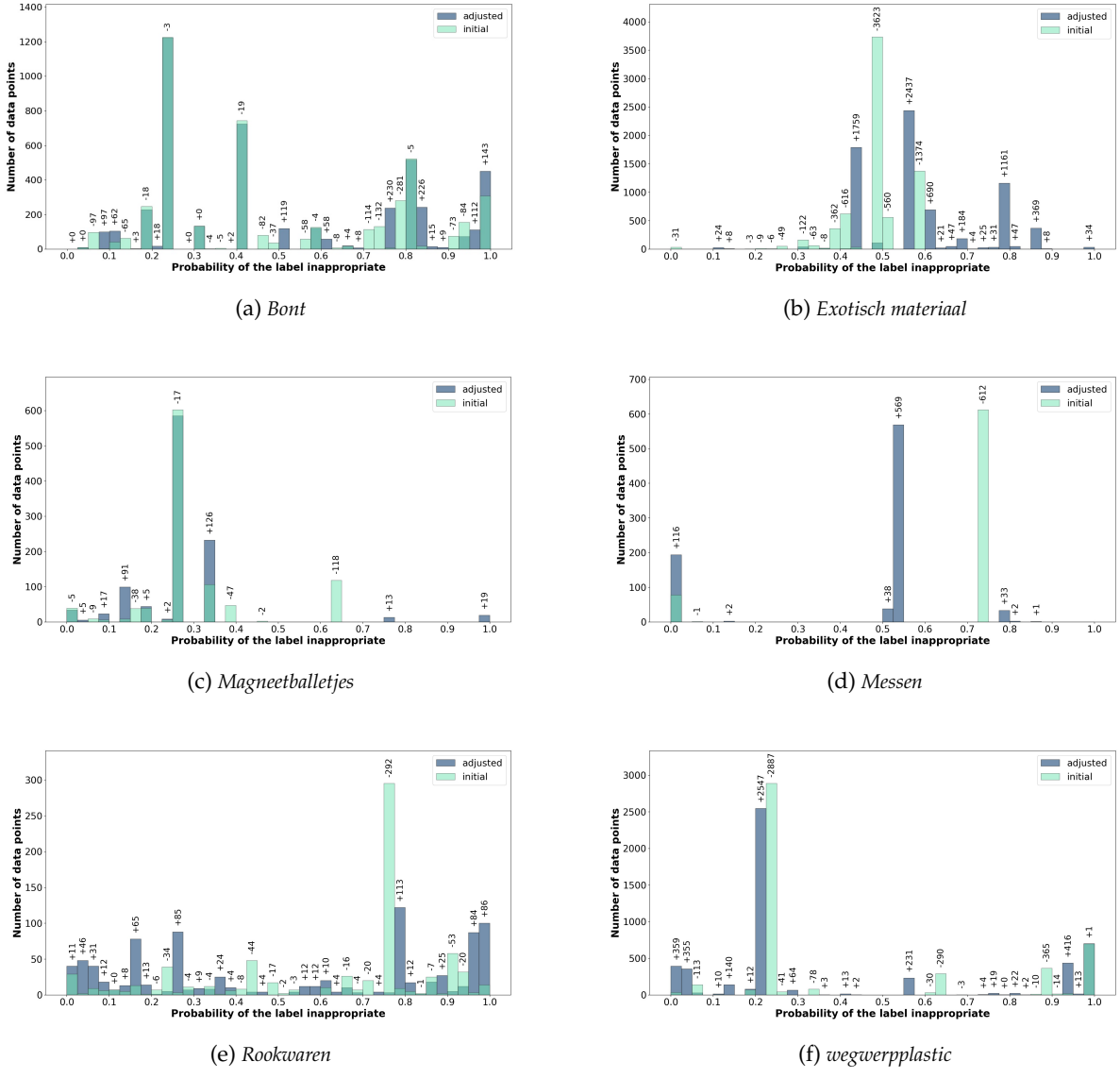
(f) *wegwerpplastic*

Figure 5.10: The change in fractions of the training sets that received a certain number of labels per sample between using the initial and the adjusted LFs.

For every monitor, Figure 5.10 shows which fraction of the training set received how many labels per sample.
In all monitors, the amount of unlabeled data decreased to a certain degree, while the sizes of the fractions with higher numbers of labels increased. The most significant

increase of this kind could be perceived in *messen* for the fraction with two labels. Overall, this monitor was the one with the biggest changes in fraction sizes.
In the monitors *exotisch materiaal* and *messen*, most labeled samples received a label from at least two LFs. In all other monitors, most labeled samples received exactly one label.

5.2.3.2 Probability of the label inappropriate



(a) *Bont*

(b) *Exotisch materiaal*

(c) *Magneetballetjes*

(d) *Messen*

(e) *Rookwaren*

(f) *wegwerpplastic*

Figure 5.11: The change in the distribution of probabilities for the label *inappropriate* over the training samples as suggested by the label model trained on the whole set of LFs for each monitor. The change is shown between using the initial and the adjusted set of LFs. The probabilities are divided into 40 bins of size 0.025 each.

For every monitor, Figure 5.11 shows the distribution of the probabilities of the label *inappropriate* across the training samples in a histogram. Altogether, the assigned probabilities for the label *inappropriate* are almost placed within the full range between 0 and 1 in all monitors (except for *messen*, where the highest probability was close to 0.9). In

the monitor *messen*, the label model output only 7 different probability values across training samples.
In all monitors except for *messen*, the tendencies of the label models stayed the same, and the bins shifted more towards the ends of the possible value range. In *messen*, instead of rather high values, the label model output mostly values close to 0.50 after the adjustments. Also, the number of samples with very low probabilities increased in this monitor.

5.2.3.3 LFs' statistics and effects on the label model

Performance \ Monitor	*bont*	*exotisch materiaal*	*magneetballetjes*	*messen*	*rookwaren*	*wegwerpplastic*
Recall	0.98	1.00	0.71	0.39	0.58	0.60
Precision	0.40	0.06	0.63	0.07	0.09	0.56
F2	0.76	0.23	0.70	0.20	0.27	0.59
Recall (initial)	0.98	0.00	0.57	0.11	0.67	0.60
Precision (initial)	0.42	0.00	0.17	0.02	0.10	0.54
F2 (initial)	0.78	0.00	0.39	0.06	0.31	0.59

Table 5.6: The performance of the label model on the validation set using the complete set of LFs across all monitors.

Table 5.6 shows the performance of the label model on the validation set when all LFs were used before (initial) and after the adjustments. We present this table to allow for comparisons with individual LFs' effects on the label model's performance, which were also calculated on the validation set. For each LF, Table 5.7 shows the statistics, how it had been changed, as well as both the effects on the label model before and after the adjustments. For newly added LFs, there were no effects before the adjustments.
In comparison to their initial performance, the label models in *exotisch materiaal*, *magneetballetjes* and *messen* showed a clear performance increase. In *rookwaren* there was a clear decrease in recall and a small decrease in precision. In the other monitors, there were only small changes.
Overall, the effects of most LFs that had initially had zero coverage remained unchanged (except for LF 0 from *magneetballetjes*). Also, in most LFs whose effect on the performance had initially been less than 1.00%, there was no perceivable change of this effect. In all monitors, the domain experts had adjusted existing LFs. In these LFs, negative effects on the label models' performance strongly increased (e.g. LF 0 from *rookwaren*), while positive effects on the performance showed a clear decrease (e.g. LF 3 from *magneetballetjes*). Instead of an effect smaller than 1.00%, the inclusion of LF 1 from *rookwaren* had a strong negative effect on the performance after the adjustments. The only LF with a stronger positive effect on the performance after the adjustments was LF

0 from *magneetballetjes*. In LF 1 from *messen*, the effect on the performance stayed the same.
In all monitors, the domain experts had added new LFs. The effect of most of these LFs on the performance was below 1.00%. Most of the remaining LFs had a strong effect on the performance (a positive one in LF 7 from *exotisch materiaal* and LFs 8 and 9 from *messen*; a negative one in LF 17 from *rookwaren* and LF 7 from *messen*).
There were also LFs that had not been adjusted. A lot of these LFs' effects only slightly changed. For the others, it varied across monitors how their effects were affected. In *bont* and *magneetballetjes*, previous effects on the performance of vanished or considerably decreased. In *rookwaren*, several of these LFs that had previously had a small positive effect on the performance had now developed a strong negative one. In *exotisch materiaal* and *messen*, there were LFs that developed a strong positive effect through the adjustments.

Index	Change	Polarity	Coverage	Overlaps	Conflicts	Correct	Incorrect	%Effect on LM			% Effect on LM (initial)		
								Recall	Precision	F2	Recall	Precision	F2
0	/	[0]	0.25	0.09	0.04	77	1	0.00	0.00	0.00	0.00	3.03	1.14
1	/	[0]	0.01	0.01	0.01	3	0	0.00	0.00	0.00	0.00	-1.59	-0.58
2	/	/	0.00	0.00	0.00	0	0	0.00	0.00	0.00	0.00	0.00	0.00
3	/	[0]	0.02	0.01	0.00	6	0	0.00	0.00	0.00	0.00	0.00	0.00
4	/	[0]	0.03	0.02	0.01	9	3	0.00	0.00	0.00	0.00	0.00	0.00
5	/	[0]	0.01	0.00	0.00	3	0	0.00	0.00	0.00	0.00	0.00	0.00
6	/	[0]	0.21	0.12	0.07	79	5	0.00	0.00	0.00	0.00	3.03	1.14
7	A	[0, 1]	0.02	0.02	0.01	3	6	3.70	-4.98	0.61	3.70	-1.03	2.01
8	/	[1]	0.00	0.00	0.00	0	0	0.00	0.00	0.00	0.00	0.00	0.00
9	A	[1]	0.13	0.10	0.06	11	34	1.85	-30.54	-8.27	5.56	-27.25	-4.34
10	A	[1]	0.03	0.03	0.01	7	3	0.00	-3.91	-1.44	0.00	-0.79	-0.29
11	/	[1]	0.00	0.00	0.00	4	0	0.00	0.00	0.00	0.00	0.00	0.00
12	/	[1]	0.00	0.00	0.00	4	0	0.00	0.00	0.00	0.00	0.00	0.00
13	/	[0, 1]	0.16	0.10	0.06	45	49	68.52	21.00	59.29	70.37	25.64	61.95
14	/	/	0.00	0.00	0.00	0	0	0.00	0.00	0.00	0.00	0.00	0.00
15	N	[1]	0.00	0.00	0.00	0	0	0.00	0.00	0.00	/	/	/
16	N	[1]	0.01	0.01	0.00	2	0	0.00	0.00	0.00	/	/	/
17	N	[0]	0.00	0.00	0.00	0	0	0.00	0.00	0.00	/	/	/

(a) Bont

Index	Change	Polarity	Coverage	Overlaps	Conflicts	Correct	Incorrect	%Effect on LM			Effect on LM (initial)		
								Recall	Precision	F2	Recall	Precision	F2
0	A	[0, 1]	0.73	0.67	0.66	10	206	70.00	-956.00	-44.00	-0.50	-0.02	-0.09
1	/	[1]	0.00	0.00	0.00	2	0	0.00	0.00	0.00	0.00	0.00	0.00
2	/	[1]	0.10	0.10	0.10	7	32	0.00	-2.33	-1.89	-1.00	-0.04	-0.19
3	/	[0]	0.02	0.02	0.02	10	0	0.00	-0.57	-0.47	-1.00	-0.05	-0.21
4	A	[0]	0.00	0.00	0.00	0	0	0.00	0.00	0.00	-1.00	-0.05	-0.20
5	/	[0]	0.83	0.67	0.64	242	0	0.00	22.47	19.10	-1.00	-0.04	-0.19
6	/	[0]	0.11	0.10	0.08	39	2	0.00	0.00	0.00	-1.00	-0.05	-0.19
7	N	[0]	0.08	0.08	0.07	54	0	0.00	22.81	19.40	/	/	/
8	N	[1]	0.22	0.22	0.21	7	5	0.00	-0.57	-0.47	/	/	/

(b) Exotisch materiaal

Index	Change	Polarity	Coverage	Overlaps	Conflicts	Correct	Incorrect	%Effect on LM			% Effect on LM (initial)		
								Recall	Precision	F2	Recall	Precision	F2
0	A	[1]	0.01	0.00	0.00	4	0	20.00	8.57	17.71	0.00	0.00	0.00
1	A	[0]	0.04	0.02	0.00	13	0	0.00	0.00	0.00	0.00	0.00	0.00
2	/	[0]	0.31	0.06	0.00	114	0	0.00	0.00	0.00	0.00	32.35	17.74
3	A	[1]	0.01	0.01	0.01	4	3	20.00	-60.00	10.00	100.00	100.00	100.00
4	A	[0]	0.14	0.04	0.00	44	1	0.00	0.00	0.00	0.00	0.00	0.00
5	/	[0]	0.02	0.02	0.01	6	0	0.00	0.00	0.00	0.00	0.00	0.00
6	N	[1]	0.00	0.00	0.00	0	0	0.00	0.00	0.00	/	/	/

(c) Magneetballetjes

Index	Change	Polarity	Coverage	Overlaps	Conflicts	Correct	Incorrect	%Effect on LM			% Effect on LM (initial)		
								Recall	Precision	F2	Recall	Precision	F2
0	/	/	0.00	0.00	0.00	0	0	0.00	0.00	0.00	0.00	0.00	0.00
1	A	[0]	0.49	0.46	0.45	96	2	100.00	100.00	100.00	100.00	100.00	100.00
2	/	[1]	0.00	0.00	0.00	0	0	0.00	0.00	0.00	0.00	0.00	0.00
3	/	/	0.00	0.00	0.00	0	0	0.00	0.00	0.00	0.00	0.00	0.00
4	/	[0]	0.04	0.01	0.00	11	0	28.57	18.14	22.80	0.00	2.04	1.18
5	A	[0]	0.05	0.04	0.00	12	0	28.57	-810.71	-55.36	0.00	0.00	0.00
6	N	[0]	0.00	0.00	0.00	0	0	0.00	0.00	0.00	/	/	/
7	N	[0]	0.12	0.06	0.01	25	0	28.57	-810.71	-55.36	/	/	/
8	N	[1]	0.03	0.01	0.01	5	3	71.43	69.32	70.23	/	/	/
9	N	[1]	0.45	0.45	0.45	0	83	71.43	69.00	70.05	/	/	/

(d) Messen

Index	Change	Polarity	Coverage	Overlaps	Conflicts	Correct	Incorrect	%Effect on LM			% Effect on LM (initial)		
								Recall	Precision	F2	Recall	Precision	F2
0	A	[0, 1]	0.42	0.27	0.25	18	55	-28.57	-134.29	-79.72	0.00	-51.85	-27.45
1	A	[1]	0.02	0.02	0.01	2	1	-14.29	-12.91	-13.41	0.00	0.00	0.00
2	/	[1]	0.11	0.10	0.05	6	13	0.00	0.00	0.00	0.00	0.00	0.00
3	/	[0]	0.05	0.05	0.04	5	2	-28.57	-24.03	-25.67	-25.00	-12.64	-16.91
4	/	[0]	0.09	0.06	0.05	15	0	-14.29	-12.91	-13.41	0.00	6.82	4.41
5	/	[0]	0.04	0.04	0.03	8	1	0.00	2.38	1.52	0.00	2.38	1.52
6	/	[0]	0.02	0.01	0.00	6	0	0.00	1.20	0.76	0.00	1.20	0.76
7	/	[0]	0.04	0.03	0.03	3	0	0.00	-1.23	-0.78	0.00	3.53	2.26
8	/	[0]	0.03	0.02	0.02	2	1	0.00	0.00	0.00	0.00	0.00	0.00
9	/	[0]	0.00	0.00	0.00	3	0	0.00	0.00	0.00	0.00	0.00	0.00
10	/	[0]	0.08	0.07	0.06	16	0	0.00	0.00	0.00	0.00	4.65	2.99
11	/	[0]	0.00	0.00	0.00	1	0	0.00	0.00	0.00	0.00	0.00	0.00
12	/	[0]	0.01	0.01	0.00	1	0	0.00	0.00	0.00	0.00	0.00	0.00
13	/	[1]	0.01	0.01	0.01	2	3	-14.29	-12.91	-13.41	-12.50	-11.14	-11.64
14	/	[1]	0.01	0.01	0.00	1	1	0.00	0.00	0.00	0.00	0.00	0.00
15	/	[1]	0.17	0.16	0.14	4	33	-28.57	-40.57	-35.89	12.50	-1.06	4.41
16	N	[0]	0.02	0.00	0.00	0	0	0.00	0.00	0.00	/	/	/
17	N	[0]	0.21	0.18	0.18	41	2	-42.86	-50.18	-47.39	/	/	/
18	N	[0, 1]	0.17	0.09	0.04	8	21	0.00	3.53	2.26	/	/	/

(e) Rookwaren

Index	Change	Polarity	Coverage	Overlaps	Conflicts	Correct	Incorrect	%Effect on LM			% Effect on LM (initial)		
								Recall	Precision	F2	Recall	Precision	F2
0	/	[1]	0.02	0.01	0.00	17	1	8.45	3.11	7.37	8.45	3.32	7.38
1	A	[0]	0.07	0.01	0.00	32	2	2.82	-31.30	-2.85	2.82	-25.77	-2.26
2	A	/	0.00	0.00	0.00	0	0	0.00	0.00	0.00	0.00	0.00	0.00
3	A	/	0.00	0.00	0.00	0	0	0.00	0.00	0.00	0.00	0.00	0.00
4	A	[0]	0.00	0.00	0.00	1	0	0.00	0.00	0.00	0.00	0.00	0.00
5	A	[0]	0.46	0.11	0.00	167	1	8.45	-52.98	-0.07	8.45	-54.93	-0.54
6	A	[0]	0.09	0.07	0.00	44	0	0.00	5.22	1.16	0.00	2.94	0.66
7	/	[0]	0.01	0.01	0.00	5	0	0.00	0.00	0.00	0.00	-1.54	-0.33
8	/	[0]	0.01	0.01	0.00	0	0	0.00	0.00	0.00	0.00	0.00	0.00
9	/	[0]	0.01	0.00	0.00	1	1	1.41	0.63	1.24	1.41	0.66	1.24
10	A	[0, 1]	0.01	0.01	0.00	4	2	0.00	-1.60	-0.34	0.00	-2.33	-0.50
11	/	[0]	0.06	0.03	0.00	21	4	4.23	-8.60	1.77	4.23	-11.88	1.11
12	/	[0]	0.00	0.00	0.00	0	0	0.00	0.00	0.00	0.00	0.00	0.00
13	/	[1]	0.07	0.01	0.00	58	9	66.20	40.38	62.78	66.20	42.05	62.81
14	N	[0]	0.01	0.01	0.00	3	0	0.00	0.00	0.00	/	/	/
15	N	[0]	0.05	0.04	0.00	16	0	0.00	0.00	0.00	/	/	/

(f) Wegwerpplastic

Table 5.7: For each LF in each monitor, this table shows the change, i.e. whether it was adjusted (A), newly added (N) or unchanged (/). Further, the table shows the LFs' statistics on the training set (polarity, coverage overlaps, conflicts), the number of incorrect and correct suggestions on the validation set and their effect on the label model's (LM) performance before (initial) and after the adjustments. Gray lines indicate LFs with zero coverage.

5.2.4 Performance on the test data

Session, model \ Monitor		*bont*	*exotisch materiaal*	*magneetballetjes*	*messen*	*rookwaren*	*wegwerpplastic*
initial	label model	**0.80**	0.02	0.08	0.03	0.31	0.57
	classifier	0.77	0.18	0.00	0.27	0.26	0.80
set 1	label model	0.78	0.19	0.49	0.03	0.31	0.64
	classifier	0.68	0.14	0.47	0.22	0.27	0.75
set 2	label model	0.78	0.00	**0.65**	0.03	**0.36**	0.57
	classifier	0.71	0.00	0.61	**0.27**	0.26	**0.85**
set 3	label model	0.77	**0.24**	0.59	0.17	0.23	0.57
	classifier	0.70	0.20	0.55	0.18	0.11	0.81

Table 5.8: Performance (F2 score) of the label model and the classifier (classifier) for all monitors on the test sets across the monitor creation and improvement sessions. The initial monitor creation session is referred to as *initial*, while improvement sessions are referred to by indices for the inspiration set type that was used (*set 1* stands for previously unlabeled data, *set 2* stands for data on which LFs previously disagreed, *set 3* stands for data most mistaken by the classifier). For each monitor, the best-performing model across all sessions is marked in bold.

Tables 5.8 shows the F2-scores of the label models (using all LFs) and the classifiers (and thus the whole pipeline) for the monitor creation (initial) or the monitor improvement sessions (based on a certain inspiration set type). Performance of the whole pipeline denotes the performance of the classifier (relying on the feature group combination that had been best on the validation data), i.e. the last step in the pipeline.
In most monitors (*magneetballetjes*, *messen*, *rookwaren* and *wegwerpplastic*), the best performance across sessions was obtained using the adjusted LF sets from the improvement session that relied on the inspiration set 2. Notably, in the monitor *exotisch materiaal*, the scores of both the label model and the whole pipeline were zero using adjustments inspired by this set. Instead, the best score was obtained using the adjustments from the improvement session that used the inspiration set 3. The monitor *bont* was the only one in which the best score was obtained in the monitor creation session without any inspired adjustments.

Strikingly, there were only two monitors (*messen* and *wegwerpplastic*) in which the best score was obtained by the whole pipeline. In all other monitors, a label model had the best score and thus made better predictions on the test data than any classifier.

5.3 Discussion

> **RQ3a:** ***How do LF adjustments inspired by fast inspection of previously unlabeled data change the LFs' effect on the label models' performance?***

In most monitors, the domain experts increased the coverage of the LF set. This coverage increase was the strongest across all inspiration set types. After their adjustments, LFs seemed to be more correlated in which data points they labeled. In most of the monitors, their adjustments made the label model suggest that there were data points with a high probability of being appropriate, but also data points with a high probability of being inappropriate. In three monitors, the label model's tendencies in assigned probability values changed.
Regarding the label model's performance on the validation data using all LFs, we saw different effects across monitors. In two monitors, the label model performed slightly worse. In the others, performance increased with emphasis on recall in three monitors and on precision in the other.
The majority of individual LFs showed only little change in effect on the performance after the adjustments.

> **RQ3b:** ***How do LF adjustments inspired by fast inspection of data on which LFs previously disagreed change the LFs' effect on the label models' performance?***

In half of the monitors, the domain experts slightly increased the coverage of the LF set. Also LFs seemed to be more correlated in which data points they labeled in these monitors. In *exotisch materiaal*, we perceived the opposite effect.
In all monitors, their adjustments made the label model suggest that there were data points with a high probability of being appropriate, but also data points with a high probability of being inappropriate. At the same time, the label model in *exotisch materiaal* developed a tendency towards suggesting that most data points were appropriate.
The adjustments affected performance of this label model in various ways across monitors. In two monitors, performance did not change. In two other monitors, we saw small performance decreases. In the other two monitors, the adjustments strongly increased recall or precision.
After the adjustments, the majority of individual LFs that had not been changed showed only small changes in their effects on the performance. In adjusted LFs, positive influences on the label model's performance seemed to decrease. At the same time, some LFs that had previously had negative effects on the performance developed a positive one after their adjustment.

> **RQ3c:** ***How do LF adjustments inspired by fast inspection of data most mistaken by the classifier change the LFs' effect on the label models' performance?***

In all monitors, the domain experts increased the coverage of the LF set. After their adjustments, LFs seemed to be more correlated in which data points they labeled.
In most monitors, their adjustments made the label model suggest that there were data points with a high probability of being appropriate, but also data points with a high probability of being inappropriate. In two monitors, the label model's tendencies in assigned probability values changed.
With respect to that label model's performance, we observed that the adjustments led to a clear increase in two monitors and a clear decrease in one. In the other monitors,

performance changes were only small.
In LFs that were not adjusted, the majority had only little changes in their effects on performance. The effect changes across the remaining LFs that were not adjusted varied across monitors. In most LFs that were adjusted, we perceived a strong increase of negative effects and and a clear decrease of positive effects on the performance.

Across all inspiration set types, we note the following insights:

- After the adjustments, most monitors still contained individual LFs which negatively affected the performance of the label model.
- Changes to some LFs might heavily affect how other LFs influence the label model's performance. This was especially obvious in LFs that were not adjusted. For instance, LF 1 from *messen* had completely different effects on the performance while not being adjusted. After adjustments based on previously unlabeled data, this LF had a small increase in conflicts and a heavily negative effect on the performance. After adjustments based on data on which LFs previously disagreed, the same LF did not show any change in conflicts or its effect on the performance.
- Using *previously unlabeled data* as an inspiration helped the most in increasing the coverage of the LF set. Intuitively, this makes sense, since that inspiration set type shows the user those data points which are currently not covered by the LFs.

With regard to previous work, our inspiration sets' effects on the coverage could be due to our static way of drawing these sets. Recall that the LF set was iteratively updated in [9]: In each iteration, one data point was drawn (with a certain strategy) using the LF set that had been updated in the previous iteration. Cohen-Wang et al. note that in early iterations, their disagreement-based strategy (i.e. selecting the data points with the maximum number of LF disagreements) drew points that were not covered by LFs yet. On real-world data, they did not perceive great differences between the coverage of their disagreement-based strategy and their abstain-based strategy throughout iterations. We, by contrast, drew our complete inspiration set once since our users could not translate their rules into LFs themselves. Aiming to increase diversity in the inspiration sets, we also did not draw data points according to the maximum number of disagreements. This was because we had noticed that the same constellation of LFs was always responsible for the most disagreements in the label matrix. It seems reasonable to assume that dynamically drawing data points like in [9] could have helped increase the coverage using *data on which LFs disagreed*.

> **RQ4:** ***Which of our three inspiration set types is best-suited for improving the performance of our whole pipeline in a short amount of time?***

Overall, the idea of improvements induced by inspiration sets seemed to work in most monitors. This was particularly evident in monitors whose F2 score had initially been below 0.10, but showed a strong increase after adjustments.
Across all improvement sessions, the removal of LFs happened only once. This is in line with [35]'s statement that most LFs encode supervision sources that are too useful to remove. The session in which the LF was removed was using *data on which LFs previously disagreed* to inspire adjustments and the removed LF originated from *exotisch materiaal*. This was the monitor in which the conflicts among LFs seemed to have impaired the performance of the whole remaining pipeline in the initial experiment. Looking at data points with disagreements, it seems reasonable that the domain expert identified one LF that was too contradictory to the others and removed it.

Most monitors seemed to have their best performance after the improvements made in the session that used data on which LFs previously disagreed. Interestingly, this was also the case for *messen*, for which this inspiration set consisted of only four data points. This suggests that compared to the ones in the other inspiration sets, these data points were particularly useful to improve this monitor.
The inspiration set with data previously unlabeled by LFs led to clear, but not the highest performance improvements in the monitors. Similarly, data on which the classifier had been most mistaken also inspired useful, but apparently not the most effective adjustments to LF sets (except for those from *exotisch materiaal*). This suggests that using a supervised inspiration set does not necessarily inspire more useful improvements than using an unsupervised one.
Surprisingly, the performance of *bont* was not improved by any adjustments based on inspiration sets. In consequence, we assume that whether or not a certain inspiration set type helps is highly dependent on a monitor. We further assume that none of our strategies to draw an inspiration set met the needs of that monitor. Another surprising observation was that data on which LFs previously disagreed led to zero performance in *exotisch materiaal*, although this had been the monitor in which the number of conflicts initially seemed to impair the performance. The following are possible explanations for this phenomenon:

- On average, the time in an improvement session was only enough to inspect thirty products. This suggests that the full potential of the inspiration sets probably was not used, as there was the chance that some important properties of the products inside them was not discovered.
- The product moderators often already had improvements to the LFs in mind after only taking a look at the rules they had designed without inspecting the data. As they worked with the categories on a daily basis, this could be an effect of the additional domain knowledge gained in the time in between the creation of a monitor and its improvement. In some cases, the domain experts invested a considerable amount of time scanning through the products in the inspiration sets, trying to find justifications for these changes. This means that some changes they made might not have been inspired by the inspiration sets, but rather by the LFs themselves.
- We told domain experts to change the set of LFs based on the datasets and also informed them how this dataset had been drawn, but we didn't limit them to only making changes that only corresponded with this way of drawing the dataset. It was therefore possible for them to also reduce LFs' contradictions while inspecting data previously unlabeled or data that had been most mistaken by the classifier.
- While the order of the inspiration sets with previously unlabeled data or data on which LFs previously disagreed was randomized in the inspiration sessions, data most mistaken by the classifier had always been shown at last due to time constraints. Hence, the domain experts were more experienced in designing LF adjustments during the session that used data most mistaken by the classifier. It is possible that this experience influenced the quality of their improvements.

Across both the monitor creation and the monitor improvement sessions' results, the classifier mostly performed worse than the label model in a monitor. The original pipeline design provides that the classifier's output should always be used to create the suggestions that the product moderator would judge on. Based on these results, it seems more reasonable to decide which model should be used based on a small hand-labeled set of data samples for each monitor. In order not to overfit to such a set,

samples that are newly identified as inappropriate by the experts when using the tool could be appended to this set.

6 | General Discussion

In this chapter, we discuss insights and findings with respect to our requirements and challenges we identified in Chapter 3 and reflect on limitations of our approach that might have caused unexpected results.

The first nonfunctional requirement had been that the domain experts can formulate a monitor's underlying logic independently of technical expert by expressing their domain knowledge. Our pipeline was designed in a way that keeps all parts but those in which the domain experts define the logic fixed. Yet, the product moderators can only independently create or adjust a monitor if they are able to completely write LFs themselves. In the present research, we functioned as a bridge between the domain knowledge and the pipeline by translating the product moderators' rules into Python functions. To fully estimate the usefulness of our pipeline, it is necessary to eliminate this human building bridge (e.g. by creating LF templates as in [7]).

The second nonfunctional requirement had been that a monitor can be quickly created at a reasonable level of performance. In our initial experiments, promising results in two monitors proved that this was possible. However, the other monitors performed poorly. The following aspects of our experiments might have caused this poor performance:
We had fixed limits on the amount of time domain experts used to create or improve the sets of LFs. It is possible that the domain experts were not able to cover all data points whose properties could have helped the classifier in learning proper representations of both classes. Most previous work on Snorkel did not mention time limits. In [38], a time limit was used, but users evaluated LFs' errors during their LF definition.
That we did not enable the users to evaluate their LFs while defining them is another aspect that could have negatively affected the LFs and thereby the labels on which the classifier was trained. Several works using Snorkel had given domain experts more time to design such functions, or let them evaluate their LFs during the definition (e.g. [10], [13], [34]).
Also, our pipeline design determined that the whole set of LFs should be used to label the data that the classifier should be labeled on. However, in our experiments, including the whole set of LFs sometimes led to lower performance, whereas leaving only one LF out resulted in a significant performance rise. Based on this finding, we suggest that the LF ablation analysis should become a fixed component in the pipeline. This would ensure that the best combination of LFs would be taken for labeling the training data and probably prevent zero performance values in the discriminative classifiers.

The third nonfunctional requirement had been that a monitor can be quickly adjusted to improve performance. Our experiments indicate that inspiration sets are a promising way to guide such improvements in a small amount of time in most monitors. In our experiments, using a supervised inspiration set did not show advantages over using an

unsupervised inspiration set in most monitors. Instead, the most promising approach had been to use data on which LFs had conflicted.

One of the main challenges we had identified had been that the monitoring system should be able to generalize beyond the target property description provided by the domain expert. Strikingly, across the results of all sessions, the label models alone mostly seemed to produce better results than those of the whole pipeline. Considering that generalizability is considered as the main strength of including the classifier in the Snorkel architecture [23], this appears unexpected at first. However, the way we set up our experiments could have influenced this phenomenon.
Besides the points we raised above, we used the same simple classifier throughout our experiments. It could be the case that our classifier is a limitation and that others would have yielded better results. Likewise, the choice of features could have been a limitation. For instance, we had not included products' images as a source of information for our classifier. By contrast, the product moderators told us that these images were an important source of information for them while producing the gold labels. Also, by letting the domain experts influence the set of features considered in a monitor, we injected another type of domain knowledge than most previous work. It is also possible that the information on which the features were based had flaws. For instance, numerical or categorical attribute values often were not filled out. Also, we realized that not all texts were completely written in Dutch, but e.g. in English.
Another limitation of our approach is the amount of data we used. It lies in the nature of the product moderation categories at hand that their scope varies, and that some of them have only a small set of products of interest that they should be applied to. However, the differences between the amount of training data and the amount of gold-labeled sets showed great variations across monitors. Therefore, some validation or test sets might have a more representational sample of a category's scope than others. Further, it has been shown that the performance of the classifier in the data programming pipeline improves with increasing number of unlabeled data (which then gets labeled by the label model) [26]. It could thus help to collect data (including products that later get deleted) over time to obtain larger training sets. Another option would be data augmentation.
Besides the dataset sizes, the individual data samples cause a limitation. Manual inspection revealed that for some categories, different colors (e.g. for single-use plastic items) or sizes (e.g. of clothes) were included in the datasets. In consequence, some products could have been over-represented in the data. We had made sure to filter out duplicate offers of exactly the same products, but we had not computed the similarity between different samples and filtered those out that were too similar.
Another source of possible limitations are the product moderators. The first limitation is their general experience with defining rules. Whereas one product moderator had been mainly responsible for writing the Boolean queries in their currently used monitoring tool, the others were mainly responsible for validating potentially inappropriate products. The level of experience in query writing could affect their ability to define heuristic rules.
Another limitation is that they showed different approaches to the rule improvement task. The product moderator who wrote the most queries directly edited a rule when obtaining a new idea. Another product moderator first wrote all new indicators for a label that they had encountered in an inspiration set on paper and created the rule improvements as a last step. The third product moderator indicated having fully understood the principle of rule definition after the second improvement session.

Also, one domain expert was responsible for three and another for two monitors, while the other expert was responsible for only one monitor. In some cases, the product moderators had reported that the true labels of some products had been questionable for a small amount of products. This mostly occurred in items on which the information provided on the product page was too sparse. As human resources available for this research were limited and one product moderator usually was the expert for a certain category, we received gold labels by one expert only, and therefore had no means to compute inter-annotator agreements on the labels.
Another limitation are the categories themselves. In this research, we investigated only six of them, whereas the actual number of categories of inappropriate items that arise from bol.com's platform policy is considerably larger. The results might therefore not generalize to all product moderation categories for the platform.

Another main challenge for a monitoring system had been that the reasoning behind a monitor's logic should be easy to understand. After the last session in the monitor improvement experiments, we asked the product moderators for their impression of our prototype. They perceived the division between the definition of a monitor's scope and the definition of that monitor's target property as helpful. Also, they found that expressing a monitor's target property across separate rules that each covered different supervision signals increased clarity. In consequence, we regard this challenge as successfully met.

The third main challenge for a monitoring system had been that this system should be able to rank candidate products by their probability of being inappropriate. The evaluation of such a ranking was not the object of the present research. However, for a side project, a product moderator ranked items suggested by the monitor *wegwerpplastic*. They used the monitor version after adjustments based on data on which LFs had disagreed, since that version had yielded the best performance on the test set. The product moderator focused on those products that were suggested to be most severe. Consequently, they ranked products according to the confidence of the classifier for the label *inappropriate*. They reported that they were satisfied with these suggestions. This indicates that our monitoring system is able to provide them with the ranking they need.

7 | Future Work

Based on our results and the limitations of the study we conducted, we suggest the following directions for future work:

The results of this research indicate that the performance of the label model in the Snorkel pipeline might be sensitive to small changes of an LF. In consequence, a possible direction for follow-up studies is the investigation of when it is advisable to combine LFs into one and when it is better to separate high-coverage LFs into single LFs covering smaller logical units (e.g. sub-conditions).
Another promising direction could be to systematically vary prior assumptions about data imbalance (as present in our datasets) and measure their effect on the pipeline's performance.
Also, to enable more general assumptions about the usefulness of individual LFs' improvements, one could conduct a change ablation study, investigating the effect of a single LF's change on the label model. Such a study could provide valuable insights to inform more efficient LFs improvements.
In our results, we recognized that leaving out some LFs had a strong effect on the performance of the label model, while leaving out others did not seem to affect the performance at all. An analysis that could produce valuable insights would be an ablation study in which different numbers of LFs are left out at a time. Such an LF group ablation study could help determine sets of LFs that are particularly useful for a given task, and those that might hinder performance. The properties of such LFs could be used to further inform LF definition guidance.

Another line of future research could repeat our experiments using an updated pipeline design. For example, one could add a fixed LF ablation component to use the LF sets that produce the best-performing label model on a validation set for labeling training data. Also, instead of having a fixed threshold for calculating the performance scores, one could investigate ways to select an individual threshold for each monitor based on the tendencies of the probabilities for the label *inappropriate* that the label model assigned to training data.
In this study, we relied on the product moderators' domain knowledge about a category for creating an initial set of LFs. Instead, future work could focus on different ways of drawing an initial development set. For instance, different proportions between inappropriate and appropriate products could be systematically varied. Another idea is to select development sets based on the difference of one possible source of information by e.g. including products with maximally different descriptions or drawing a certain amount of samples from offers clustered by product images.
Also, it seems promising to investigate how well the pipeline performs when different

groups of domain experts with different levels of coding experience define LFs. A respective study could present the same fictional (to exclude familiarity effects) categories as tasks to domain experts. Then, the study could examine differences of the created monitors between the groups.

Future work could also invent different types of inspiration sets. For instance, a possible inspiration set could be formed from data points at which the difference model in Socratic learning [34] detected the greatest difference between the label model and the classifier.
In this setting, we set time limits to improvements. The time spent on a task had been decided according to business logic. Our findings from chapter one might allow for creating a first set of LF templates. Using these templates, follow-up work could try to examine what amount of time is actually needed to reach a good level of performance for different monitors. Templates would allow non-coder domain experts to formulate LFs that could be directly translated into Python functions. This would enable a dynamic way of drawing inspiration sets, as well as measuring improvements over time.
Furthermore, it could be insightful to change the type of problem investigated to find out how well-suited the idea of inspiration sets is for different problem types. For instance, future work could use the pipeline for high-precision problems, for problems on a different corpus or for similar high-recall problems in a different domain.

8 | Conclusion

The present research constitutes a first step towards informing the design of a data programming-based system that can support product moderators by suggesting potentially inappropriate products.

In Chapter 4, we focused on setting up recall-oriented monitors in our pipeline for individual product moderation categories. We aimed to identify the main characteristics of LFs that were based on domain experts' domain knowledge and created in a short amount of time. In most monitors, the domain experts seemed to express the main indicators as high-coverage LFs and add numerous low-coverage LFs targeted at specific cases. Our results suggest that designing a high-coverage set of LFs, as well as high-accuracy LFs targeting positive cases was challenging under these conditions. Furthermore, during the LF definition, domain experts also seemed to consider cases that were not present in the data, leading to LF with zero coverage.
We investigated the effects of these LFs on the performance of the label model in the data programming pipeline. The general performance of this model seemed to reflect the level of experience the responsible domain expert had with that category. Not all LFs had a positive effect on the label model's performance. Most individual LFs' effects on the performance were small. A small amount of LFs had unexpected effects on the label model. It seems reasonable to attribute these effects to the interplay of these LFs with all others in the set for a monitor. Since in one monitor, LFs' contradictions seemed to impair performance, domain experts should be advised to watch the amount of conflicts.
We evaluated the performance of the whole monitor pipeline after LF definition. In the majority of monitors, performance was poor. Overall, the label models outperformed the classifier.

In Chapter 5, we focused on improving the monitors in a short amount of time using *inspiration sets* (intelligently drawn sets of data points to inspire LF improvements). We extracted two unsupervised (*previously unlabeled data* and *data on which LFs previously disagreed*) inspiration sets and one supervised (*data most mistaken by the classifier*) inspiration set for each monitor.
We investigated how adjustments inspired by these inspiration sets affected the LFs' effects on the label model's performance and listed tendencies across monitors. In some monitors, the adjustments let the label model change its tendency towards assigning low or high probabilities of being inappropriate to products. These tendencies are indicative of whether the adjustments improve the label model's performance. Overall, adjusted LFs seemed more correlated in which data points they covered. The LF sets' coverage increased and this increase was the highest after adjustments based on previously unlabeled data. This means that an inspiration set based on previously unlabeled

data was most helpful in overcoming the initial challenge of writing a high-coverage set of LFs. After adjusting LFs, these LFs' effects on the performance mostly showed little changes. The effects of some single LFs made a greater change. Whether or not the adjustments to a monitor were successful cannot be directly deduced from the direction of the changes of these effects. Care should still be taken while adjusting monitors, since we observed that changes to some LFs can influence LFs that were not touched. We also aimed to identify which inspiration set was best suited to improve the monitors' performance in a short amount of time. Again, we observed that the label model was not outperformed by the classifier. The strategy that yielded the best results in four out of six monitors was using data on which LFs previously disagreed. This suggests that using a supervised type of inspiration set does not necessarily lead to great advantages over using an unsupervised set. Still, our results indicate that there was not a one-size-fits-all solution: Different monitors seemed to respond differently to adjustments based on the inspiration set types.

In the beginning of this thesis, we argued that data programming allowed product moderators to steer classifiers by adjusting the automatic creation of training data. It is important to note that the classifier often performed worse than the label model. Likewise, a better label model performance did not necessarily lead to a better classifier performance. This suggests that controlling the training data creation did not give the product moderators full responsibility over the classifier's decisions. Future research is necessary to investigate how this responsibility could be realized.

Bibliography

[1] Alonso, O., Rose, D. E., and Stewart, B. Crowdsourcing for relevance evaluation. *SIGIR Forum 42* (2008), 9–15.

[2] Arnold, P., Wartner, C., and Rahm, E. Semi-automatic identification of counterfeit offers in online shopping platforms. *Journal of Internet Commerce 15*, 1 (2016), 59–75.

[3] Bach, S. H., He, B., Ratner, A., and Ré, C. Learning the structure of generative models without labeled data. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70* (2017), ICML'17, JMLR.org, p. 273–282.

[4] Bach, S. H., Rodriguez, D., Liu, Y., Luo, C., Shao, H., Xia, C., Sen, S., Ratner, A., Hancock, B., Alborzi, H., Kuchhal, R., Ré, C., and Malkin, R. Snorkel drybell: A case study in deploying weak supervision at industrial scale. In *Proceedings of the 2019 International Conference on Management of Data* (New York, NY, USA, 2019), SIGMOD '19, Association for Computing Machinery, p. 362–375.

[5] bol.com. Verkooppartners bouwen letterlijk met bol.com aan platform van toekomst. `https://pers.bol.com/alle-persberichten/verkooppartners-bouwen-letterlijk-met-bol-com-aan-platform-van-toekomst/`, 2019. [Online; accessed 28-03-2020].

[6] bol.com. Wat zijn de richtlijnen en beleid op assortiment versie september 2019. `https://partnerplatform.bol.com/help/aanbieden/is-mijn-aanbod-geschikt/assortimentsvoorwaarden`, 2019. [Online; accessed 28-03-2020].

[7] Bringer, E., Israeli, A., Shoham, Y., Ratner, A., and Ré, C. Osprey: Weak supervision of imbalanced extraction problems without code. In *Proceedings of the 3rd International Workshop on Data Management for End-to-End Machine Learning* (New York, NY, USA, 2019), DEEM'19, Association for Computing Machinery.

[8] Callahan, A., Fries, J. A., Ré, C., Huddleston, J. I., Giori, N. J., Delp, S. L., and Shah, N. H. Medical device surveillance with electronic health records. *NPJ Digital Medicine 2* (2019).

[9] Cohen-Wang, B., Mussmann, S., Ratner, A., and Ré, C. Interactive programmatic labeling for weak supervision. In *KDD '19: 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Workshop on Data Collection, Curation, and Labeling (DCCL) for Mining and Learning, August 05, 2019, Anchorage, AK.* (2019).

[10] Dunnmon, J. A., Ratner, A. J., Saab, K., Khandwala, N., Markert, M., Sagreiya, H., Goldman, R., Lee-Messer, C., Lungren, M. P., Rubin, D. L., and Ré, C. Cross-modal data programming enables rapid medical machine learning. *Patterns 1*, 2 (2020), 100019.

[11] Dutta, P., and Saha, S. A weak supervision technique with a generative model for improved gene clustering. *2019 IEEE Congress on Evolutionary Computation (CEC)* (2019), 2521–2528.

[12] Dutta, P., Saha, S., Pai, S., and Kumar, A. A protein interaction information-based generative model for enhancing gene clustering. *Scientific Reports 10*, 1 (2020), 1–12.

[13] Ehrenberg, H. R., Shin, J., Ratner, A. J., Fries, J. A., and Ré, C. Data programming with ddlite: Putting humans in a different part of the loop. In *Proceedings of the Workshop on Human-In-the-Loop Data Analytics* (New York, NY, USA, 2016), HILDA '16, Association for Computing Machinery.

[14] Fries, J. A., Varma, P., Chen, V. S., Xiao, K., Tejeda, H., Saha, P., Dunnmon, J., Chubb, H., Maskatia, S., Fiterau, M., et al. Weakly supervised classification of aortic valve malformations using unlabeled cardiac mri sequences. *Nature communications 10*, 1 (2019), 1–10.

[15] Grave, E., Bojanowski, P., Gupta, P., Joulin, A., and Mikolov, T. Learning word vectors for 157 languages. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)* (2018).

[16] Hancock, B., Varma, P., Wang, S., Bringmann, M., Liang, P., and Ré, C. Training classifiers with natural language explanations. *Proceedings of the conference. Association for Computational Linguistics. Meeting 2018* (2018), 1884–1895.

[17] Köpcke, H., Thor, A., Thomas, S., and Rahm, E. Tailoring entity resolution for matching product offers. In *Proceedings of the 15th International Conference on Extending Database Technology* (New York, NY, USA, 2012), EDBT '12, Association for Computing Machinery, p. 545–550.

[18] Mackey, T., Kalyanam, J., Klugman, J., Kuzmenko, E., and Gupta, R. Solution to detect, classify, and report illicit online marketing and sales of controlled substances via twitter: using machine learning and web forensics to combat digital opioid access. *Journal of medical Internet research 20*, 4 (2018), e10029.

[19] Mackey, T. K., and Kalyanam, J. Detection of illicit online sales of fentanyls via twitter. *F1000Research 6* (2017).

[20] Mintz, M., Bills, S., Snow, R., and Jurafsky, D. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2* (2009), Association for Computational Linguistics, pp. 1003–1011.

[21] Nashaat, M., Ghosh, A., Miller, J., and Quader, S. Wesal: Applying active supervision to find high-quality labels at industrial scale. In *Proceedings of the 53rd Hawaii International Conference on System Sciences* (2020).

[22] Nashaat, M., Ghosh, A., Miller, J., Quader, S., Marston, C., and Puget, J.-F. Hybridization of active learning and data programming for labeling large industrial datasets. In *2018 IEEE International Conference on Big Data (Big Data)* (2018), IEEE, pp. 46–55.

[23] Ratner, A., Bach, S. H., Ehrenberg, H. R., Fries, J. A., Wu, S., and Ré, C. Snorkel: Rapid training data creation with weak supervision. *Proceedings of the VLDB Endowment. International Conference on Very Large Data Bases 11 3* (2017), 269–282.

[24] Ratner, A., Hancock, B., Dunnmon, J., Goldman, R., and Ré, C. Snorkel metal: Weak supervision for multi-task learning. In *Proceedings of the Second Workshop on Data Management for End-To-End Machine Learning* (New York, NY, USA, 2018), DEEM'18, Association for Computing Machinery.

[25] Ratner, A., Hancock, B., Dunnmon, J., Sala, F., Pandey, S., and Ré, C. Training complex models with multi-task weak supervision. In *Proceedings of the AAAI Conference on Artificial Intelligence* (2019), vol. 33, pp. 4763–4771.

[26] Ratner, A. J., De Sa, C. M., Wu, S., Selsam, D., and Ré, C. Data programming: Creating large training sets, quickly. In *Advances in Neural Information Processing Systems 29*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, Eds. Curran Associates, Inc., 2016, pp. 3567–3575.

[27] Ratner, A. J., Hancock, B., and Ré, C. The role of massively multi-task and weak supervision in software 2.0. In *CIDR* (2019).

[28] Ré, C., Niu, F., Gudipati, P., and Srisuwananukorn, C. Overton: A data system for monitoring and improving machine-learned products. *arXiv preprint arXiv:1909.05372* (2019).

[29] Roh, Y., Heo, G., and Whang, S. E. A survey on data collection for machine learning: A big data - ai integration perspective. *IEEE Transactions on Knowledge and Data Engineering* (2019), 1–1.

[30] Saab, K., Dunnmon, J., Goldman, R., Ratner, A., Sagreiya, H., Ré, C., and Rubin, D. Doubly weak supervision of deep learning models for head ct. In *International Conference on Medical Image Computing and Computer-Assisted Intervention* (2019), Springer, pp. 811–819.

[31] Saab, K. K., Dunnmon, J., Ré, C., Rubin, D. L., and Lee-Messer, C. Weak supervision as an efficient approach for automated seizure detection in electroencephalography. *NPJ Digital Medicine 3* (2020).

[32] Stonebraker, M., and Ilyas, I. F. Data integration: The current status and the way forward. *IEEE Data Eng. Bull. 41*, 2 (2018), 3–9.

[33] Suiter, K., and Sferrazza, S. Monitoring the sale and trafficking of invasive vertebrate species using automated internet search and surveillance tools.

[34] Varma, P., He, B., Iter, D., Xu, P., Yu, R., De Sa, C., and Ré, C. Socratic learning: Correcting misspecified generative models using discriminative models. *arXiv preprint arXiv:1610.08123* (2017).

[35] Varma, P., Iter, D., De Sa, C., and Ré, C. Flipper: A systematic approach to debugging training sets. In *Proceedings of the 2nd Workshop on Human-In-the-Loop Data Analytics* (New York, NY, USA, 2017), HILDA'17, Association for Computing Machinery.

[36] Varma, P., and Ré, C. Snuba: automating weak supervision to label training data. *Proceedings of the VLDB Endowment 12*, 3 (2018), 223–236.

[37] Varma, P., Sala, F., He, A., Ratner, A., and Re, C. Learning dependency structures for weak supervision models. In *International Conference on Machine Learning* (2019), pp. 6418–6427.

[38] Wu, S., Hsiao, L., Cheng, X., Hancock, B., Rekatsinas, T., Levis, P., and Ré, C. Fonduer: Knowledge base construction from richly formatted data. In *Proceedings of the 2018 International Conference on Management of Data* (2018), pp. 1301–1316.

[39] Xu, Q., Li, J., Cai, M., and Mackey, T. K. Use of machine learning to detect wildlife product promotion and sales on twitter. *Frontiers Big Data 2* (2019), 28.